한 권으로 배우는
파이썬 기초 & 알고리즘 사고법

Python and Algorithmic Thinking for the Complete Beginner
by Aristides S. Bouras and Loukia V. Ainarozidou

한권으로 배우는
파이썬 기초 & 알고리즘 사고법

1쇄 발행 2018년 7월 10일

지은이 아리스티데스 보우라스, 루키아 아이나로지두
옮긴이 길준민, 임종범, 송의성, 유헌창
펴낸이 장성두
펴낸곳 주식회사 제이펍

출판신고 2009년 11월 10일 제406-2009-000087호
주소 경기도 파주시 회동길 159 3층 3-B호
전화 070-8201-9010 / **팩스** 02-6280-0405
홈페이지 www.jpub.kr / **원고투고** jeipub@gmail.com
독자문의 readers.jpub@gmail.com / **교재문의** jeipubmarketer@gmail.com

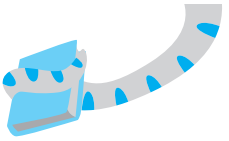
편집부 이종무, 황혜나, 최병찬, 이 슬, 이주원 / **소통·기획팀** 민지환 / **회계팀** 김유미
교정·교열 안종군 / **본문디자인** 성은경 / **표지디자인** 미디어픽스
용지 에스에이치페이퍼 / **인쇄** 한승인쇄 / **제본** 광우제책사

ISBN 979-11-88621-20-0 (93000)
값 30,000원

한 권으로 배우는 파이썬 기초 & 알고리즘 사고법

Python and Algorithmic Thinking for the Complete Beginner:

Learn to Think Like a Programmer



Aristides Bouras, Loukia Ainarozidou 지음
길준민, 임종범, 송의성, 유헌창 옮김

Jpub 제이펍

※ 드리는 말씀

- 이 책에 기재된 내용을 기반으로 한 운용 결과에 대해 저자, 역자, 소프트웨어 개발자 및 제공자, 제이펍 출판사는 일체의 책임을 지지 않으므로 양해 바랍니다.
- 이 책에 기재한 회사명 및 제품명은 각 회사의 등록 상표(또는 상표)이며, 본문 중에는 ™, ©, ® 등의 기호를 생략하고 있습니다.
- 이 책에서 사용하고 있는 실제 제품 버전은 독자의 학습 시점에 따라 책의 버전과 다를 수 있습니다.
- 책 내용과 관련된 문의사항은 옮긴이나 출판사로 연락 주시기 바랍니다.

  - 옮긴이: joonmin.gil@gmail.com
  - 출판사: readers.jpub@gmail.com

배우는 것을 즐기고,

그럼으로써 자신의 목표에

꼭 도달하기를 바랍니다.

차 례

PART V 루프 제어 구조

옮긴이 머리말

길준민

2016년 1월 스위스 다보스에서 열린 '세계 경제 포럼'에서 '제4차 산업 혁명'이라는 주제가 논의되면서 전 세계는 제4차 산업 혁명 시대에 대비하기 위한 변화의 기로에 서 있습니다. 제4차 산업 혁명 시대에는 인공지능, 3D 프린팅, 자율주행차, 생명공학 등과 사물 인터넷(IoT), 클라우드, 빅 데이터 등이 결합하여 우리에게 더 풍요로운 미래를 제공해 줄 것으로 예상하고 있습니다.

그러나 최근의 화두는 '제4차 산업 혁명 시대를 사는 인간은 어떤 역할을 해야 하는가?'입니다. 이세돌 9단이 인공지능 바둑 기계와의 대결에서 4:1로 패배함으로써 앞으로 인공지능이 인간을 지배할 것이라는 의견이 대두되고 있습니다. 하지만 인공지능 바둑 기계의 학습 데이터로 사용되는 기보는 어떻게 만들어진 것인지를 생각해 봐야 합니다. 인공지능 바둑 기계는 바로 인간이 수천 년 동안 축적해 놓은 기보 데이터를 단지 기계적으로 학습하고 그것을 바둑에 적용한 것입니다. 인간이 축적해 놓은 기보 데이터가 없었다면 인간과 대결할 정도의 인공지능 바둑 기계는 탄생하지 못했을 것입니다. 결국, 인간 없는 인공지능 바둑 기계는 무용지물인 셈입니다. 인간이 수천 년 동안 축적해 놓은 기보 데이터는 단숨에 이루어진 것이 아니라 세대를 거치면서 만들어진 산물이라 할 수 있습니다. 이러한 점에서 볼 때 창의적 문제 해결 능력은 기계가 흉내 낼 수 없는 인간 역량 중 하나이고, 세대를 거치면서 모든 분야에 걸쳐 혁신을 이끌어 내는 능력이며, 제4차 산업 혁명 시대에 반드시 인간이 갖춰야 능력입니다.

스티브 잡스는 하버드 대학 연설에서 "이 나라에 살고 있는 모든 사람은 컴퓨터 프로그래밍을 배워야 합니다. 프로그래밍은 생각하는 방법을 가르쳐 주기 때문입니다"라고 주장했습니다. 빌 게이츠(마이크로소프트 창업자), 마크 저커버그(페이스북 설립자) 등과 같은 많은 오피니언 리더들 또한 이와 똑같은 견해를 갖고 있습니다.

그렇다면 생각하는 방법을 가르쳐 주는 컴퓨터 프로그래밍을 배우기 위해서는 어떤 프로그래

밍 언어를 배워야 할까요? 복잡하고 배우기 어려운 컴퓨터 프로그래밍 언어보다는 이해하기 쉽고 손쉽게 활용할 수 있으며, 다양한 분야에 적용할 수 있는 컴퓨터 프로그래밍 언어가 좋을 것입니다.

이 책에서 다루고 있는 파이썬은 어렵고 복잡하게만 느꼈던 프로그래밍 언어의 세계를 흥미롭게 탐험할 수 있는 훌륭한 프로그래밍 도구라 생각합니다. 프로그래밍은 자신이 생각하는 바를 논리적으로 구상하고 그것을 알고리즘(algorithm)으로 표현하며, 코딩(coding)으로 실행에 옮길 수 있는 과정으로, 프로그램 개발뿐 아니라 최근에는 컴퓨팅 사고(computational thinking)를 배양할 수 있는 도구로 활용되고 있습니다. 특히, 컴퓨터 전공자는 물론 비전공자라도 논리적 사고 능력과 창의적 문제 해결 능력을 배양할 수 있도록 해 주는 것이 바로 프로그래밍입니다. 이러한 의미에서 초보자도 쉽게 따라 할 수 있는 파이썬 언어야말로 자신의 생각과 논리를 표현하기에 가장 적절하고 강력한 프로그래밍 도구입니다.

한편, 이 책은 컴퓨터 비전공자도 쉽게 이해하고 활용할 수 있는 풍부한 예제를 제공함으로써 프로그래밍 기본 개념의 정립뿐 아니라 문제 해결력 증진에 도움이 되는 알고리즘 사고를 배양하는 데 중점을 두었습니다. 따라서 실생활과 밀접하게 연관된 다양한 예제를 통해 창의적 문제 해결 능력이 자연스럽게 증진될 수 있을 것입니다.

이 책은 원저자의 오랜 경험과 능력, 치밀한 구성에 의해 완성된 책이므로 완성도 및 충실도가 매우 뛰어납니다. 역자들도 평소 무심코 넘겼던 개념에 대해 다시 한 번 생각해 보는 계기를 마련해 주기도 했습니다. 그만큼 이 책은 프로그래밍 개념 정립과 알고리즘 사고 배양에 관한 바이블로서 손색이 없다고 생각합니다. 이 책을 번역하면서 원저자의 프로그래밍에 관한 해박한 지식과 식견에 놀라움을 감출 수가 없었습니다. 그래서 원저자의 생각과 의도를 여러 번 생각해 가며 이 책의 완결성에 해가 되지 않도록 최선을 다해 번역했습니다. 이 책을 프로그래밍에 관심이 있는 입문자나 프로그래밍 초보자들에게 프로그래밍의 기본 개념을 정립해 줄 입문서로서 추천합니다.

끝으로, 이 책이 나오기까지 많은 수고를 해 주신 제이펍 사장님을 비롯한 관계자 여러분께 감사드립니다. 또한, 출판 막바지까지 교정 작업을 도와준 대구가톨릭대학교 IT공학부 대학원생 이윤수, 테이퍼악떠라 학생과 고려대학교 컴퓨터학과 대학원생 명노영, 박봉우 학생의 노고에 진심으로 감사드립니다. 이 책이 독자 여러분이 파이썬 언어의 개념을 파악하고 프로그래밍 능력의 향상을 통한 알고리즘 사고의 증진과 창의적 문제 해결력을 배양하는 데 조금이나마 도움이 되길 바랍니다.

임종범

세상이 급격하게 변화함에 따라 시대가 요구하는 인재상도 바뀌었습니다. 현시대에는 기존 문제들에 대한 해답을 잘 외우고 학습한 사람보다는 새로운 문제에 직면했을 때 그 문제를 해석하고 해결 방안을 찾아낼 수 있는 사람을 원하고 있습니다. 이것은 단순히 프로그래밍 책이 아니라 여러 문제에 대한 해결 방안을 모색하고, 이를 단계별로 사고하는 데 도움을 주는 책입니다. 물론, 그 도구는 프로그래밍 언어가 될 것입니다.

만약 여러분이 어느 날 갑자기 사막에 떨어졌다고 가정해 봅시다. 여러분들은 아마도 구조를 받을 때까지 살아남아야겠다는 생각을 할 것입니다. 이러한 목표를 달성하기 위해서는 구체적인 목표를 만들어 하나하나 실천해야 합니다. 알고리즘적 사고도 이와 마찬가지입니다. 새로운 문제에 직면했을 때 이를 작은 문제들로 나누고, 이 작은 문제들을 해결해 나가면서 커다란 문제를 해결해 나가는 과정이 바로 알고리즘적 사고입니다. 그 과정들을 프로그래밍 언어를 통해 작성해 나가는 과정을 통해 여러분들은 알고리즘적 사고를 학습할 수 있습니다.

이 책은 컴퓨터 과학 분야를 전공하고자 하는 사람뿐만 아니라 프로그래밍 언어를 처음 접하는 사람, 프로그래밍을 취미로 하고 싶은 사람, 프로그래머가 어떤 방식을 통해 사고하는지 알고 싶어 하는 사람 모두에게 적합한 책입니다. 이 책을 모두 학습하고 나면 프로그래머가 얼마나 창의적이며 능동적인 사고를 하는지 알 수 있을 것입니다. 아무쪼록 이 책을 통해 여러분들이 알고리즘과 프로그래밍을 바라보는 시야가 넓어지기를 기대합니다.

송의성

파이썬은 프로그래밍 언어가 어떤 것이며 무슨 일을 할 수 있는지 분명하고 쉽게 알 수 있도록 해 주는 언어입니다. 따라서 프로그래밍 언어에 익숙하지 않은 초보자들이 처음 접하는 교육용 언어로 널리 사용되고 있으며, 활용 가치도 높아 다양한 분야의 실무에 적용되고 있습니다. 이러한 측면에서 볼 때 파이썬 초보자에게 쉽고 다양한 경험을 제공하고, 사고력을 키워 주는 훌륭한 교재의 제공은 매우 중요합니다. 그러나 지금까지는 단순히 문제의 결과를 얻기 위해 교재의 코드를 입력해 보고 그 결과를 확인해 보는 수준의 단순 코딩 작업 위주의 교재가 많았고, 알고리즘에 기반을 둔 문제 해결력을 향상시킬 수 있는 교재는 거의 없었습니다.

역자는 처음 이 책을 접했을 때 알고리즘에 기반을 둔 파이썬을 이토록 쉽고 자세하게 그리고 다양한 종류의 많은 예제를 다루고 있는 것에 놀라움을 금할 수 없었습니다. 그동안 출간되었던 초보자용 파이썬 책들에서 부족해 보여 아쉬웠던 점들이 거의 완벽히 채워져 있던 책이었

습니다. 특히, 초보자들이 힘들어하는 알고리즘적 사고력을 신장시키는 데 많은 도움을 줄 수 있다는 생각이 들었습니다. 우리나라에는 왜 이런 책이 없을까 하는 아쉬움도 있었지만, 이제 부족하나마 번역서가 출간되니 프로그래밍 언어 초보 입문자들에게 큰 도움이 되었으면 하는 바람입니다.

마지막으로, 이 책을 집필한 저자에게 감사하며 책을 번역하는 동안 많은 힘이 되어 준 가족들에게도 감사를 전합니다. 그리고 이 책이 완성될 때까지 많은 협조와 지원을 해 주신 출판사 관계자 여러분들께도 감사드립니다.

유헌창

애플의 전 CEO인 스티브 잡스는 "소프트웨어와 컴퓨터가 세상의 중심이 돼 가면서 코딩에 대한 지식이 기술 중심의 경제 사회에서 장점이 된다"라고 언급했습니다. 미국의 전 대통령인 오바마 또한 "코딩은 개인의 미래가 아니라 국가의 미래를 위해 중요하다"라고 말했습니다. 코딩을 배우는 것은 컴퓨팅 사고력을 비롯하여 창의성을 키워 줄 수 있다는 사실을 많은 연구 결과가 증명하고 있습니다. 결국, 코딩이라는 것은 컴퓨터 프로그래밍 언어를 도구로 사용해 문제를 해결하는 과정을 서술하는 것이고, 이러한 도구는 C, 자바, 파이썬 등 너무나 많은 프로그래밍 언어들이 존재합니다. 그중에서도 최근 들어 파이썬에 관련된 교육이 학교 현장에서 점차 증가하고 있는 이유는 기존 언어들과는 달리 사람이 생각하는 방식을 똑같이 표현할 수 있고, 문법이 쉬워 빠르게 배울 수 있다는 장점을 갖고 있기 때문입니다. 영어, 중국어 등과 같은 자연어와 마찬가지로 프로그래밍 언어도 쉽게 배울 수 있어야 합니다.

좋은 책을 만드는 것은 어려운 작업이지만, 기존에 만들어진 좋은 책을 번역하는 것도 어려운 작업이라 생각합니다. 원저자의 뜻을 최대한 반영하려고 노력했지만, 아직 부족한 점이 많다는 것도 번역하는 과정에서 느꼈습니다. 이 책을 번역하는 과정에서 많은 부분을 수정 및 보완했지만, 오류가 있을 수 있습니다. 독자 여러분들의 많은 관심으로 책의 완성도가 높아지기를 기대해 봅니다.

저자 소개

아리스티데스 보우라스

Aristides[1] S. Bouras는 1973년에 태어났다. 어린 시절부터 컴퓨터 프로그래밍을 좋아했으며, 12살에 ROM 기반 버전의 BASIC 프로그래밍 언어와 64킬로바이트의 RAM이 통합된 코모도어 64를 첫 번째 컴퓨터로 소유했다.

피레에푸스(Piraeus)의 기술교육 연구소에서 컴퓨터 엔지니어링 학위, 트라키아(Thrace)의 Democritus Polytechnic 대학교에서 전기 및 컴퓨터공학 학위를 받았다. 산업 데이터 흐름과 제품 라벨링에 특화된 회사에서 소프트웨어 개발자로 근무했으며, 마이크로소프트 SQL 서버에 데이터를 수집하고 저장하는 PC 소프트웨어 애플리케이션과 데이터 터미널용 소프트웨어 애플리케이션을 주로 개발하였다. 기업용 웨어하우스 관리 시스템 등의 다양한 애플리케이션을 개발하였으며, 지금은 고등학교에서 교사로 근무하며 주로 컴퓨터 네트워크, 인터넷/인트라넷 프로그래밍 툴 및 데이터베이스 과목을 가르치고 있다. 공저자인 루키아 아이나로지두와 결혼해 두 명의 자녀를 두고 있다.

루키아 아이나로지두

Loukia V. Ainarozidou는 1975년에 태어났다. 13세의 나이에 128킬로바이트의 RAM과 내장형 3인치 플로피 디스크 드라이브를 장착한 Amstrad CPC6128을 첫 번째 컴퓨터로 소유했다.

---

1 아리스티드(기원전 530~468년)는 고대 아테네의 정치가이자 장군이었다. 고대 역사학자 헤로도토스(Herodotus)는 그를 '아테네에서 가장 훌륭하고 존경스러운 인물'이라고 기술했다. 그는 그가 한 모든 일에서 매우 공평했기 때문에 종종 '아리스티드의 정의'라고 불렸다. 그는 아테네의 고전 시대 초반에 활약하였고, 아테네인들이 살라미스와 플라타의 전투에서 페르시아인들을 물리치도록 도와주었다.

피레에푸스의 기술교육 연구소에서 컴퓨터 엔지니어링 학위, 트라키아의 Democritus Polytechnic 대학교에서 전기 및 컴퓨터 공학 학위를 받았다. 과일 및 채소의 포장을 담당하는 회사의 데이터 물류 부서에서 감독관으로 일했으며, 지금은 고등학교에서 교사로 근무하며 컴퓨터 네트워크, 컴퓨터 프로그래밍 및 디지털 디자인 과목을 가르치고 있다. 공저자인 아리스티데스 보우라스와 결혼해 두 명의 자녀를 두고 있다.

감사의 글

야니 카포스(Yannis T. Kappos) 박사가 없었다면 이 책을 출간하지 못했을 것이다. 오토캐드 기술 서적의 유명 저자인 그는 우리가 이 책을 출간할 수 있도록 격려해 주었고, 우리의 모든 질문, 심지어는 바보 같은 질문에도 답변을 위해 많은 시간을 할애해 주었다. 편집에 도움을 준 우리의 친구이자 수석 편집자인 빅토리아 오스틴(Victoria(Vicki) Austin)에게도 감사의 인사를 전한다. 그녀가 없었다면 이 책이 가진 잠재력을 완전히 발휘하지 못했을 것이다. 그녀의 수고와 가치 있고 건설적인 제안으로 이 책의 수준이 한층 높아질 수 있었다.

이 책의 구성 방법

이 책은 미국의 심리학자 제롬 브루너(Jerome Bruner)가 1960년에 제안한 나선형 교육 과정의 교수 방법을 따른다. 이 방법에 따르면, 독자가 학습 주제를 완전히 이해할 때까지 학습 주제를 배울 때 매번 보다 정교한 수준으로 기본 아이디어가 주기적으로 재검토된다. 먼저, 독자는 세부 사항에 신경 쓰지 않고 기본 요소들을 학습한다. 이후에 더 많은 세부 사항의 내용을 학습하게 되고 이때에도 기본 요소가 반복적으로 언급되어, 결국 뇌의 장기 기억 저장소에 저장된다.

제롬 브루너에 따르면, 학습은 학생의 적극적인 참여, 실험, 탐구 및 발견이 필요하다. 이 책은 많은 예제를 포함하고 있으며, 예제 대부분은 실제 생활에 활용될 수 있다. 이를 통해 독자는 파이썬을 사용하여 자신의 프로그램을 만들 수 있다.

이 책의 대상 독자

컴퓨터 프로그래밍을 배우고 싶지만 컴퓨터 프로그래밍을 전혀 모르는 사람들을 위한 책이다. 이 책이 애플릿이나 데스크톱 또는 모바일용 애플리케이션을 만드는 법을 가르쳐 줄 것인지를 궁금해한다면 그 대답은 '아니요'이다. 이런 목적을 위해서는 다른 책을 찾아야 할 것이며, 수많은 책이 파이썬, C# 또는 자바를 이용해서 그런 기술을 가르쳐 줄 수 있다. 그러나 그 많은 책들

중 어떤 책들은 24시간 안에 가르쳐 줄 수 있다고 주장한다! 웃기는 이야기다! 이런 책들이 그렇게 해 줄 수도 있을지는 모르지만, 그렇게 하기 위해서는 모두가 당연하게 생각하는 것 중 한 가지를 전제 조건으로 해야 할 것이다. 즉, 독자가 컴퓨터 프로그래밍에 대한 기본 사항들은 이미 알고 있다는 가정 말이다. 불행히도 그런 책들 중 어느 것도 초보 프로그래머가 배워야 할 첫 번째 항목인 '알고리즘적 사고(Algorithmic Thinking)'를 가르쳐 주지 않는다.

알고리즘적 사고는 단순히 코드를 배우는 것 이상으로 코드 작성법을 학습하는 과정까지 포함한 문제 해결 과정을 배우는 것이다. 800페이지가 넘는 본문에 300개 이상의 프로그래밍 예제와 400개 이상의 프로그래밍 연습문제, 450개 이상의 참/거짓 문제, 150개 이상의 객관식 문제, 180개 이상의 복습 문제(인터넷에서 해결책과 답을 찾을 수 있음)가 포함된 이 책은 학생, 교사, 교수, 초보자 또는 일반 프로그래머, 적절한 규칙과 기법을 사용하여 컴퓨터 프로그래밍을 배우거나 가르치기를 원하는 사람에게 적합하다.

복습문제와 연습문제의 해답

모든 복습문제에 대한 해답과 연습문제의 해결책은 인터넷을 통해 무료로 제공된다. 다음 주소에서 해답과 해결책을 내려받을 수 있다.

★ https://bit.ly/2lAIofb(혹은 제이펍 홈페이지의 이 책 소개 페이지)

오타 신고 방법

교재의 정확성을 높이기 위해 많은 노력을 기울였지만 실수가 있을 수 있다. 이 책의 텍스트나 코드에서 오류를 발견하면 신고해 주기 바란다. 그렇게 해 주면 이 책의 다음 버전 개선에 큰 도움이 된다. 오탈자를 발견하면 다음 주소를 방문하여 알려 주면 된다.

★ http://www.bouraspage.com/report-errata(혹은 readers.jpub@gmail.com)

오탈자인 것으로 판명되면 신고 내용이 채택되고, 해당 오탈자는 웹 사이트에 업로드 후 기존의 정오표 목록에 추가된다.

이 책에 사용된 규칙

이 책에서 사용된 규칙에 대해 설명한다. 여기서 '규칙'은 텍스트의 특정 부분이 표시되는 표준 방법을 말한다.

파이썬 명령문

이 책은 파이썬 언어로 작성된 많은 예제를 사용한다. 파이썬 명령문은 다음과 같은 글체자로 표시된다.

```
This is a Python statement
```

세 점(...) 줄임표

명령문의 일반 형식에서는 예제의 목록에서 '줄임표'라고도 불리는 세 개의 점(...)을 볼 수 있다. 줄임표는 명령문의 일부는 아니지만, 목록에서 원하는 만큼의 항목을 가질 수 있음을 의미한다. 예를 들어, 일반 형식의 줄임표를 다음 명령문과 같이 표현할 수 있다.

```
display_messages ( arg1, arg2, ... )
```

위 명령문은 목록에 두 개 이상의 인자가 포함될 수 있음을 의미한다. 이와 같은 형식의 명령문을 프로그램에서 다음과 같이 사용할 수 있다.

```
display_messages ( message1, "안녕하세요!", message2, "안녕!" )
```

대괄호

일부 명령문이나 함수의 일반 형식에서는 [ ](대괄호)가 사용될 수 있다. 이 경우, 대괄호로 둘러싸인 부분은 선택 사항(옵션)을 의미한다. 예를 들어, 일반 형태의 다음 명령문에서 [, step]은 생략될 수도 있음을 의미한다.

```
range( initial_value, final_value [, step ] )
```

다음 두 명령문의 결과는 서로 다르지만, 양쪽 모두 구문적으로는 올바르다.

```
range(0, 10)
range(0, 10, 2)
```

어두운 헤더

이 책의 대부분 예제는 다음과 같은 글자체로 표시된다.

file_29_2_3

```
a = 2
b = 3

c = a + b

print(c)
```

맨 위에 있는 어두운 헤더 file_29_2_3은 프로그램을 테스트하기 위해 열어야 하는 파일 이름을 나타낸다. 이 헤더를 포함하는 모든 예제는 인터넷상의 다음 주소에서 무료로 내려받을 수 있다.

★ https://bit.ly/2lAIofb(번역서의 예제는 https://github.com/Jpub/Python_Algorithms)

안내문

이 책에서는 개념을 더 잘 이해할 수 있도록 '주목할 것!'이라는 이름의 안내문을 자주 사용한다. 안내문의 형식은 다음과 같다.

이 모양은 쪽지를 나타낸다.

이미 알려 준 것이나 기억해야 할 것

이 책은 매우 자주 이미 배운 것(아마도 이전 장에서)을 다시 기억할 수 있도록 도와준다. 기억해야 할 것들에 대한 주의를 환기시키는 알림 형식은 다음과 같다.

이 모양은 기억해야 할 것을 나타낸다.

베타리더 후기

구민정(SK주식회사)

프로그래밍 기초부터 알고리즘 사고법까지를 책 한 권에 잘 녹여낸 좋은 책입니다! 풍부한 예제와 연습문제를 제공하고 있으며, 설명 또한 매우 친절하면서도 재미있습니다. :) 실제 자주 접하는 상황을 예시로 활용하고 있고, 점점 더 나은 코드로 발전시켜 가는 과정을 보여 주고 있습니다. 책에서 제시한 문제들은 가능하면 풀어 보시기를 추천합니다. 자신의 답안과 비교하고 고민하며 풀어 나간다면 실력이 쑥쑥 향상될 것입니다.

김진영(프리랜서)

실제 현업에서 일하면서는 퀵 정렬이나 버블 정렬 등과 같은 것보다는 현실의 데이터와 상황을 반영한 알고리즘적 사고가 더 필요했습니다. 그런데 지금까지의 책들은 '이런 알고리즘이 있고, 이렇게 구성된다'와 같은 원론적인 부분이라 이론적 알고리즘과 현업에서의 알고리즘 사이에서의 괴리감을 적잖이 느꼈었습니다. 그런데 이번 베타리딩을 하면서 이 책이 그 괴리감을 완충시킬 수도 있겠다는 생각을 했습니다. 현실과 연관된 데이터 기반의 문제들이기에 보다 더 현실감 있는 알고리즘으로 느껴졌습니다.

김정헌(BTC)

파이썬으로 프로그래밍을 처음 배우고 싶은 사람에게 어울리는 책입니다. 이 책은 파이썬의 모든 기능을 설명하는 바이블은 아니지만, 프로그래밍 기본 개념인 순차 구조, 조건문, 반복문, 리스트, 함수 등을 학습하여 어떻게 프로그래밍을 하는지를 제대로 배울 수 있습니다. 그리고 자신의 실력을 테스트할 수 있는 문제가 많아 정말 혹독하게 트레이닝할 수 있습니다. 알고리즘을 배우는 과정에서 순서도와 추적표를 사용하고 있는데, 이러한 도구는 초보 프로그래머에게

큰 도움이 될 것 같습니다. 오타와 문제점이 별로 없어서 난감할 정도였고, 내용도 편집도 모두 만족스러운 책입니다. 다만, 수식과 문제가 너무 많아 초보가 다 풀고 넘어가기에는 많은 시간이 필요할 것 같습니다.

김용현(마이크로소프트 MVP)

기존의 개발 언어 입문서들은 키워드를 소개하고 이에 대한 짧은 예제와 응용 예제가 반복되는 구조가 일반적입니다. 하지만 프로그래밍이라는 것의 실체는 문제 해결의 연속입니다. 작게는 자료구조에 값을 삽입하고 이들을 구조화해 데스크와 조직, 사회적인 해결책을 내놓습니다. 이 책은 입문자를 대상으로 파이썬이라는 언어를 소개함과 동시에 순서도와 함께 문제 해결을 위한 사고의 과정을 함께 고민하게 해 줍니다. 언어와 프로그래밍을 함께 배우려는 분들에게 추천합니다. 순서도 등을 이용해 사고의 과정을 함께 고민하는 내용이 인상 깊은 책이었습니다.

한상곤(Favorie)

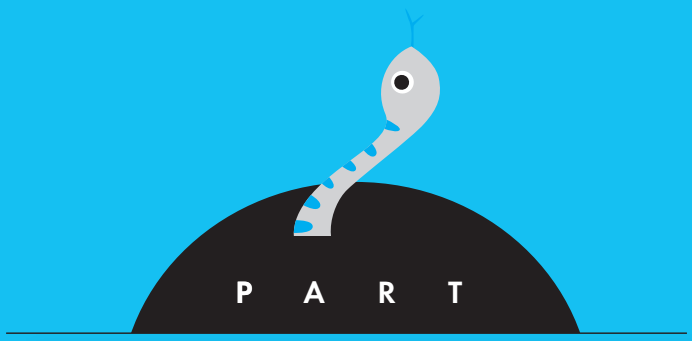
프로그래밍을 처음 시작하는 분들에게 강력하게 추천하고 싶은 책입니다. 추적표와 순서도를 사용하여 코드의 진행 과정을 한눈에 볼 수 있고, 다양한 연습문제를 제공하고 있어서 혼자서도 충분히 프로그래밍을 연습할 수 있습니다. 굳이 파이썬이 아니더라도 프로그래밍을 처음 접하는 분들에게도 좋은 책이 될 것 같습니다. 책의 절반을 '제어문'에 할애하고, 순서도나 추적표를 도입해서 코드를 논리적으로 소개하는 개념의 책이라 굉장히 좋았습니다. 가끔 대학교에서 1학년을 대상으로 파이썬을 강의해야 할 때 마땅한 교재가 없어서 너무 아쉬웠는데, 이 책이 출간되면 교재로 꼭 사용해야겠다는 생각이 들 만큼 훌륭한 내용입니다. 오타도 거의 없고 대부분의 코드를 실행했는데 별다른 오류 없이 진행되고, 코드의 양도 적당해서 좋았습니다. 주변에 프로그래밍을 처음 접하는 분들에게 이 책을 추천하고 싶네요.

한홍근(eBrain)

독자의 입장에서 고민할 법한 내용이 가득 담긴 책입니다. 문제 해결을 위한 코드를 'Why'에 근거를 두고 설명합니다. 파이썬 언어를 배우기에도 좋은 구성이지만, 코드를 작성하면서 고려해야 할 문제점과 개선법을 익히는 데에도 좋은 자료가 되리라 생각합니다. 책의 구성도 너무 재미있었습니다. 지금까지 봐 오던 책들은 이론을 설명한 후 그것으로 간단한 예제 한두 개 정도를 돌려 보는 게 일반적이었다면, 이 책은 하면 하지 말아야 할 것들과 입문자들이 실수하는 사

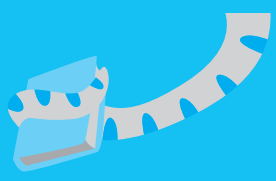
례들을 보여 주고 그걸 개선시켜 나가는, 즉 보다 독자의 입장을 고려한 구성이었습니다. 일부 어색한 문체가 조금은 있었지만 전체적인 번역 품질은 좋았습니다.

제이펍은 책에 대한 애정과 기술에 대한 열정이 뜨거운 베타리더들로 하여금
출간되는 모든 서적에 사전 검증을 시행하고 있습니다.



I

시작하기 전에



PART I 시작하기 전에

CHAPTER

01

컴퓨터의 작동 원리

1.1 들어가기

현대 사회에서는 거의 모든 업무에 컴퓨터를 사용한다. 학생들은 인터넷을 검색하거나 이메일을 보내기 위해, 직장인들은 발표 자료를 작성하거나, 데이터를 분석하거나, 고객과 소통하기 위해 컴퓨터를 사용한다. 집에서는 게임을 하거나 채팅을 하기 위해 컴퓨터를 사용한다. 아이폰과 같은 스마트폰도 이와 유사한 목적으로 사용한다. 사실, 스마트폰도 일종의 컴퓨터다.

앞에서 언급한 바와 같이 컴퓨터를 이용하여 다양한 작업을 할 수 있다. 이것이 가능한 이유는 바로 컴퓨터는 프로그램(program)에 명시된 대로 작업을 수행하기 때문이다. 여기서 프로그램이라 하는 것은 특정 작업을 수행하기 위해 컴퓨터에게 지시하는 명령문들의 집합(흔히 명령어(instruction 혹은 command)라 부른다)이다.

(일반적으로 응용 소프트웨어(application software)로 언급되는) 프로그램은 컴퓨터의 핵심이다. 프로그램이 없으면 컴퓨터는 아무런 일도 하지 않는 더미 기계(dummy machine)일 뿐이다. 실제로 프로그램은 컴퓨터에게 어떤 일을 해야 하고, 언제 수행해야 하는지를 지시한다. 이런 컴퓨터 프로그램을 설계하고, 만들고, 테스트하는 사람을 프로그래머(programmer) 혹은 소프트웨어 개발자(software developer)라고 한다.

이 책은 파이썬(Python) 언어 기반의 컴퓨터 프로그래밍에 관한 기본 개념을 소개한다.

1.2 하드웨어란?

하드웨어(hardware)는 컴퓨터를 구성하고 있는 모든 장치나 부품을 의미한다. 컴퓨터나 노트북의 케이스를 열어 보면 CPU, 메모리, 하드 디스크와 같은 여러 부품을 볼 수 있다. 컴퓨터는 장치 자체가 아니라 여러 장치가 함께 동작하는 일종의 시스템이다. 그럼, 컴퓨터 시스템의 기본 구성요소를 먼저 살펴보자.

- **중앙 처리 장치(CPU: Central Processing Unit)**
  프로그램에 정의된 모든 일(산술, 논리, 입출력 연산)을 실제로 수행하는 컴퓨터의 구성요소다.
- **주기억 장치(main memory)**
  프로그램 수행에 필요한 프로그램과 데이터를 저장하고 있는 영역이다. 주기억 장치에 저장된 모든 프로그램과 데이터는 컴퓨터를 셧다운(shutdown)하거나 플러그를 뽑으면 사라진다(즉, 휘발된다).
- **보조 기억 장치(secondary storage device)**
  일반적으로 하드 디스크(hard disk)라 하며, CD/DVD 드라이브도 이 장치에 속한다. 보조 기억 장치는 주기억 장치와 달리, 컴퓨터에 전원이 공급되지 않더라도 오랜 기간 데이터를 유지할 수 있다. 그러나 컴퓨터는 보조 기억 장치에 저장된 프로그램을 직접 실행할 수 없다. 좀 더 속도가 빠른 주기억 장치로 프로그램이 전송되고 난 후에 실행할 수 있다.
- **입력 장치(input device)**
  외부로부터 데이터를 모아 컴퓨터 내부로 들여보내는 장치를 의미한다. 키보드, 마우스, 마이크 등이 이에 해당한다.
- **출력 장치(output device)**
  데이터를 외부로 내보내기 위한 장치를 의미한다. 모니터(스크린)와 프린터가 이에 해당한다.

1.3 소프트웨어란?

컴퓨터가 수행하는 모든 것은 소프트웨어(software)가 제어한다. 일반적으로 소프트웨어는 시스템 소프트웨어(system software)와 응용 소프트웨어(application software)로 구분된다.

시스템 소프트웨어는 컴퓨터의 기본 동작을 제어하고 관리하는 프로그램이다. 예를 들어, 시스템 소프트웨어는 컴퓨터의 내부 동작을 제어하거나, 서로 연결된 모든 장치를 관리하거나,

데이터를 저장 혹은 불러들이거나, 다른 프로그램을 수행하는 역할을 한다. 윈도우, 리눅스, 맥 OS, 안드로이드, iOS와 같은 '운영체제(operating system)' 또한 시스템 소프트웨어의 한 가지 예다.

시스템 소프트웨어를 제외한 웹 브라우저, 워드프로세서, 노트패드, 게임 등과 같은 거의 모든 프로그램은 응용 소프트웨어에 해당한다.

1.4 컴퓨터가 프로그램을 어떻게 수행(실행)할까?

주기억 장치는 컴퓨터를 켰을 때 완전히 비어 있는 상태에 있다. 이때 컴퓨터는 가장 먼저 하드 디스크에서 주기억 장치로 운영체제를 전송한다.

운영체제가 주기억 장치로 전송되면 원하는 어떠한 프로그램(응용 소프트웨어)도 수행(실행)할 수 있다. 일반적으로 해당 프로그램의 아이콘을 클릭 혹은 더블클릭하거나 스마트 기기인 경우에는 손으로 눌러 실행한다. 예를 들어, 문서 편집을 하기 위해 워드프로세서 아이콘을 클릭했다고 가정해 보자. 이런 동작은 CPU가 워드프로세서 프로그램을 수행하도록 그 프로그램을 하드 디스크에서 주기억 장치로 불러들이는(혹은 복사하는) 명령을 컴퓨터에게 지시하는 것이다.

기억할 것! 프로그램은 하드 디스크와 같은 보조 기억 장치에 저장된다. 프로그램을 컴퓨터에 설치할 때 그 프로그램은 실제로 하드 디스크에 복사된다. 그러나 프로그램이 수행될 때는 하드 디스크에서 주기억 장치로 전송된 프로그램의 복사본이 실제로 수행되는 것이다.

주목할 것! '실행하다(run)'와 '수행하다(execute)'는 동의어다.

1.5 컴파일러와 인터프리터

컴퓨터는 매우 정형화된 컴퓨터 언어(computer language)로 작성된 프로그램을 수행한다. 프로그램을 영어나 한국어와 같은 자연어(natural language)를 사용하여 작성할 수는 없다. 컴퓨터는 자연어를 전혀 이해하지 못하기 때문이다.

그러면 실제로 컴퓨터는 어떤 언어를 이해할 수 있을까? 컴퓨터는 '기계어(machine language)'라는 저급(혹은 저수준) 언어(low-level language)만을 이해한다. 기계어로 작성된 모든 명령문(혹은 명령어)은 0과 1로 이루어져 있다. 다음은 기계어로 작성한 프로그램의 예로, 두 수의 합을 계산하는 프로그램이다.

```
0010 0001 0000 0100
0001 0001 0000 0101
0011 0001 0000 0110
0111 0000 0000 0001
```

위 기계어를 살펴보면 충격적이다. 그러나 걱정할 필요는 없다. 어느 누구도 위와 같은 방법으로 컴퓨터 프로그램을 작성하지는 않는다. 현시대의 모든 프로그래머는 사람이 쉽게 이해할 수 있는 고급 언어(high-level language)로 프로그램을 작성하며, 고급 언어로 작성된 프로그램을 기계어로 번역해 주는 특별한 프로그램을 사용한다. 프로그램의 번역을 위해 사용할 수 있는 프로그램으로는 컴파일러(compiler)와 인터프리터(interpreter)가 있다.

컴파일러는 고급 언어로 작성된 명령문을 개별적인 기계어 프로그램으로 번역해 주는 프로그램이다. 번역된 기계어 프로그램은 여러분이 원할 때마다 수행할 수 있다. 번역이 일단 끝나면 컴파일러는 더 이상 필요 없다.

인터프리터는 고급 언어로 작성된 명령문을 번역함과 동시에 수행하는 프로그램이다. 즉, 인터프리터는 프로그램 내의 각 명령문을 읽은 후에 그것을 기계어 코드로 번역하며, 곧바로 수행한다. 프로그램 내의 모든 명령문에 이 과정이 반복되어 수행된다.

1.6 소스 코드란?

프로그래머가 고급 언어로 작성한 명령문을 '소스 코드(source code)' 혹은 단순히 '코드(code)'라 한다. 프로그래머는 우선 소스 코드를 코드 편집기(code editor)라 불리는 프로그램을 사용하여 작성한다. 그런 다음, 해당 소스 코드를 기계어 프로그램으로 번역하기 위해 컴파일러를 사용하거나 번역함과 동시에 수행하는 인터프리터를 사용한다. 프로그래머가 소스 코드를 작성하고 실행하는 것까지를 한꺼번에 가능하게 해 주는 통합 개발 환경(IDE: Integrated Development Environment)의 한 예로 이클립스(Eclipse)가 있다. 이클립스에 관한 자세한 사항은 3장에서 학습한다.

1.7 복습문제: 참/거짓

다음 문제를 읽고 **참** 또는 **거짓**으로 답하여라.

1. 현대의 컴퓨터는 기가바이트의 주기억 장치를 가지고 있으므로 수많은 다양한 업무를 수행할 수 있다.
2. 컴퓨터는 프로그램 없이 동작할 수 있다.
3. 하드 디스크는 하드웨어의 일종이다.
4. 데이터는 전원이 없어도 오랜 기간 주기억 장치에 저장될 수 있다.
5. 데이터는 주기억 장치에 저장되지만, 프로그램은 주기억 장치에 저장되지 않는다.
6. 스피커는 출력 장치의 일종이다.
7. 윈도우와 리눅스는 소프트웨어의 일종이다.
8. 미디어 플레이어는 시스템 소프트웨어의 일종이다.
9. 컴퓨터에 전원을 켜기 전에 운영체제는 주기억 장치에 이미 존재한다.
10. 워드프로세서 애플리케이션을 열면, 그 프로그램은 보조 기억 장치에서 주기억 장치로 복사된다.
11. 기계어의 모든 명령문(명령어)은 0과 1로 이루어져 있다.
12. 현재의 컴퓨터는 0과 1을 이해하지 못한다.
13. 현재의 소프트웨어는 0과 1로 구성된 언어로 작성된다.
14. 소프트웨어는 컴퓨터의 물리적 구성요소다.
15. 고급 컴퓨터 프로그래밍 언어를 사용할 때 컴퓨터는 0과 1을 이해하지 못한다.
16. 컴파일러와 인터프리터는 소프트웨어다.
17. 컴파일러는 소스 코드를 수행 가능한 파일로 변환한다.
18. 인터프리터는 기계어 프로그램을 생성한다.
19. 소스 코드가 변환된 후에는 인터프리터가 더 이상 필요 없다.
20. 소스 코드는 텍스트 편집기를 사용하여 작성할 수 있다.
21. 컴퓨터는 컴파일 과정이나 해석 과정 없이도 소스 코드를 수행할 수 있다.
22. 기계어로 작성된 프로그램은 컴파일 과정(변환 과정)이 필요하다.
23. 컴파일러는 고급 언어로 작성된 프로그램을 기계어로 변환한다.

1.8 복습문제: 객관식

다음 문제에서 옳은 것을 골라라.

1. 다음 중 컴퓨터 하드웨어가 **아닌** 것은?
   a. 하드 디스크
   b. DVD 디스크
   c. 사운드 카드
   d. 주기억 장치
2. 다음 중 보조 기억 장치가 **아닌** 것은?
   a. DVD reader/writer 장치
   b. 하드 디스크
   c. USB 메모리
   d. RAM
3. 다음 동작 중에 CPU가 수행하지 **않는** 것은?
   a. 데이터를 주기억 장치로 이동하는 동작
   b. 데이터를 사용자에게 보여 주는 동작
   c. 주기억 장치에서 데이터를 가져오는 동작
   d. 산술 연산을 수행하는 동작
4. 터치스크린(touch screen)은 어떤 장치에 해당하는가?
   a. 입력 장치
   b. 출력 장치
   c. 위 모두 옳다.
5. 다음 중 소프트웨어가 **아닌** 것은?
   a. 윈도우
   b. 리눅스
   c. iOS
   d. 비디오 게임
   e. 웹 브라우저
   f. 위 모두 소프트웨어다.
6. 다음 설명 중 옳은 것은?
   a. 프로그램은 하드 디스크에 저장된다.
   b. 프로그램은 DVD 디스크에 저장된다.

c. 프로그램은 RAM에 저장된다.

d. 위 모두 옳다.

7. 다음 설명 중 옳은 것은?

a. 프로그램은 하드 디스크에서 직접 수행될 수 있다.

b. 프로그램은 DVD 디스크에서 직접 수행될 수 있다.

c. 프로그램은 주기억 장치에서 직접 수행될 수 있다.

d. 위 모두 옳다.

e. 위 모두 옳지 않다.

8. 다음 중 프로그래머가 컴퓨터 프로그램으로 작성할 수 **없는** 것은?

a. 기계어

b. 영어, 한국어 등의 자연어

c. 파이썬

9. 컴파일러에 대한 설명으로 옳은 것은?

a. 기계어로 작성된 프로그램을 고급 언어로 프로그램으로 변환한다.

b. 자연어(영어, 한국어 등)로 작성된 프로그램을 기계어 프로그램으로 변환한다.

c. 고급 언어로 작성된 프로그램을 기계어로 변환한다.

d. a, b, c 모두 옳지 않다.

e. a, b, c 모두 옳다.

10. 기계어에 대한 설명으로 옳은 것은?

a. 기계들끼리 소통하기 위해 사용하는 언어다.

b. 컴퓨터가 직접 사용하는 수치 명령어로 이루어진 언어다.

c. 연산을 위해 영어 단어를 사용하는 언어다.

11. 두 개의 동일한 명령문이 연속해 있다고 가정해 보자. 이때, 인터프리터의 동작 방식으로 옳은 것은?

a. 첫 번째 명령문을 번역하는 즉시 수행하고, 그런 다음 두 번째 명령문을 번역하고 즉시 수행한다.

b. 첫 번째 명령문을 번역하고, 그런 다음 두 번째 명령문을 번역한다. 그리고 이 둘을 수행한다.

c. 두 명령문이 서로 동일하기 때문에 첫 번째 명령문만을 번역하고, 그런 다음 첫 번째 명령문을 두 번 수행한다.

CHAPTER

02

파이썬

2.1 파이썬이란?

파이썬은 애플리케이션, 웹페이지 등 다양한 유형의 소프트웨어를 프로그래머가 만들 수 있는 고급 컴퓨터 프로그래밍 언어로 널리 사용되고 있다.

파이썬의 공식 홈페이지에서 자신이 **프로그래밍(programming)** 언어의 일종이라고 밝히고 있음에도 파이썬은 스크립팅(scripting) 언어로 흔히 언급되고 있다. 사실 파이썬은 이 둘 사이의 중간쯤으로, 스크립팅 언어로 사용될 수도 있고 프로그래밍 언어로도 사용될 수 있다.

2.2 스크립트와 프로그램의 차이는?

기술적으로 말하면, 스크립트는 **해석(interpret)**되는 것인 반면, 프로그램은 **컴파일(compile)**되는 것이다. 그러나 이런 설명이 이 둘 사이의 근본적인 차이점을 구별하지는 못한다. 미묘하지만 좀 더 중요한 차이점이 있다.

자바스크립트나 VBA(Visual Basic for Applications)와 같은 스크립팅 언어로 작성된 스크립트의 주요 목적은 다른 애플리케이션을 제어하는 것에 있다. 그래서 자바스크립트는 웹 브라우저를 제어하며, VBA는 MS 워드와 엑셀과 같은 마이크로소프트 오피스 애플리케이션을 제어한다.

반면, C++나 C#과 같은 프로그래밍 언어로 작성된 프로그램은 다른 애플리케이션과 완전히 독립적으로 수행된다. 이러한 프로그램은 컴파일되어 기계어 명령어들의 집합으로 만들어지고, 그런 다음 원할 때마다 단독(stand-alone)으로 수행될 수 있다.

마이크로소프트 오피스의 매크로는 VBA로 작성된 스크립트다. 이 스크립트의 목적은 마이크로소프트 오피스의 특정 함수를 자동화해 주는 것에 있다.

스크립트는 수행을 위한 호스팅 애플리케이션(hosting application)이 필요하며, 단독으로 수행될 수 없다.

2.3 파이썬을 왜 배워야 하는가?

파이썬은 '고급' 컴퓨터 언어로 알려져 있으며, 중규모 정도의 애플리케이션이나 동적 웹페이지를 개발하기에 적합한 언어다. 또한, 프로그래밍을 가르치기에 완벽한 언어이며, 과학과 수치 컴퓨팅을 위해 널리 사용되고 있다. 파이썬의 코딩 스타일은 이해하기 쉽고, 매우 효율적이다.

파이썬의 월등한 능력 중 하나는 컴퓨터의 파일 시스템과 상호작용할 수 있다는 것이다. 파이썬으로 파일을 생성할 수 있고, 파일에 쓸 수 있고, 파일에서 내용을 읽을 수도 있다. 또한, 디렉터리를 생성하거나, 파일을 지우거나, 파일 이름을 바꾸거나, 심지어 파일 속성조차 변경할 수 있다. 파이썬은 파일 시스템과 관련된 어떤 작업도 수행할 수 있어서 시스템 관리 작업에도 적합하다. 예를 들어, 파일 백업을 위한 파이썬 프로그램을 작성할 수도 있고, 파일 내용을 재포맷하는 텍스트 파일 처리용 프로그램을 만들 수도 있다.

게다가, 파이썬은 명령문이나 시스템에 설치된 다른 프로그램을 수행할 수도 있다. 그래서 C, C++, 자바 등의 다른 컴퓨터 언어로 작성되고 컴파일된 프로그램을 수행할 수 있으며, 이때 이들 프로그램의 출력 결과를 활용할 수도 있다. 이런 기능을 활용하면 여러분이 이미 작성해 놓은 프로그램을 파이썬으로 재작성하는 데 소요되는 시간을 줄일 수 있다.

수백만 혹은 수십억 줄의 코드가 이미 파이썬으로 작성되어 있다면 당연히 코드를 재사용해야겠다는 생각이 들 것이다. 이런 점에서 많은 사람이 다른 프로그래밍 언어보다 파이썬을 선호하고 있으며, 이것이 파이썬을 반드시 배워야 하는 이유다.

2.4 파이썬의 동작 방식

컴퓨터는 영어나 한국어와 같은 자연어를 이해하지 못한다. 그래서 컴퓨터와 소통하기 위해 파이썬과 같은 컴퓨터 언어가 필요하다. 파이썬은 매우 강력한 고급 컴퓨터 언어다. 파이썬 인터프리터(혹은 실제로는 컴파일러와 인터프리터의 조합)는 파이썬 언어를 컴퓨터가 실제로 이해할 수 있는 언어인 '기계어(machine language)'로 변환해 준다.

과거 컴퓨터 언어는 인터프리터나 컴파일러 중 하나를 이용하였다. 그러나 파이썬을 포함한 현재의 대다수 컴퓨터 언어는 컴파일러와 인터프리터 모두를 이용하고 있다. 파이썬 컴파일러는 파이썬 명령문을 바이트코드(bytecode) 명령문으로 번역하고, 그것을 .pyc 파일에 저장한다. 그런 다음, 인터프리터에 의해 .pyc 파일이 수행된다. 이러한 인터프리터를 '파이썬 가상 머신(Python Virtual Machine)'이라고 부르며, 이것의 임무는 바이트코드를 저급 기계어로 변환하는 것이다.

주목할 것! 파이썬 바이트코드는 파이썬 가상 머신이 수행하는 기계어다.

그림 2-1은 파이썬으로 작성된 명령문을 바이트코드로 컴파일하는 과정과 바이트코드가 어떻게 수행되는지를 보여 준다.

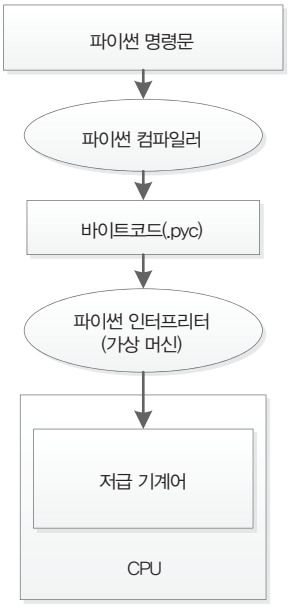


그림 2-1 파이썬 가상 머신을 이용한 파이썬 명령문의 수행 과정

이제 다음과 같은 의문이 생길 것이다. 파이썬은 왜 두 번의 변환을 할까? 파이썬 명령문을 왜 직접 저급 기계어 코드로 변환하지 않을까? 이에 대한 답변은 순전히 효율성에 있다. 오늘날 소수의 인터프리터만이 실제로 코드를 줄 단위로 직접 해석하고 수행한다. 현재 거의 모든 인터프리터는 다음과 같은 두 가지 이유 때문에 중간 과정을 거치는 방식을 사용하고 있다.

1. 중간 코드(바이트코드)가 있어야 최소한의 최적화 수행이 가능하다.
2. .py와 동일한 이름을 가진 .pyc 파일이 존재하면 파이썬은 자동으로 그 파일을 수행한다. .pyc 파일이 이미 존재하고 소스 파일을 변경하지 않았다면 소스 파일을 다시 컴파일하지 않음으로써 시간을 절약할 수 있다.

CHAPTER

08

추적표

8.1 추적표란 무엇인가?

추적표(trace table)란, 알고리즘이나 컴퓨터 프로그램을 수행하면서 발생할 수 있는 논리 오류를 검사하는 기술을 말한다. 추적표를 통해 알고리즘이나 컴퓨터 프로그램의 수행 흐름을 모의 실행(simulation)하면, 명령문들을 하나씩 실행하면서 할당 명령문에 의해 변숫값들이 어떻게 바뀌는지를 살펴볼 수 있다.

알고리즘이나 컴퓨터 프로그램이 어떻게 수행되는지를 추적표를 통해 가시적으로 보여 줄 수 있으며, 논리 오류를 검사할 수도 있다. 추적표의 일반 형태는 다음과 같다.

단계	명령문	설명	변수1	변수2	변수3
1					
2					
…					

실제로 추적표를 사용해 보자.

```
x = 10
y = 15
z = x * y
z += 1
print(z)
```

위 파이썬 코드에 있는 명령문의 각 수행 단계별 변숫값의 변화를 알아보기 위해서는 다음과 같이 추적표를 만들 수 있다.

단계	명령문	설명	x	y	z
1	x = 10	10이 변수 x에 할당됨	**10**	-	-
2	y = 15	15가 변수 y에 할당됨	10	**15**	-
3	z = x * y	변수 x와 y의 곱의 결과가 변수 z에 할당됨	10	15	**150**
4	z += 1	변수 z에 1을 더함	10	15	**151**
5	print(z)	151이 출력			

예제 8.1-1 추적표 만들기

다음 파이썬 프로그램에 대한 추적표를 만들어라.

```
Ugly = "Beautiful"
Beautiful = "Ugly"
Handsome = Ugly

print("Beautiful")
print(Ugly)
print(Handsome)
```

풀이

단계	명령문	설명	Ugly	Beautiful	Handsome
1	Ugly = "Beautiful"	문자열 "Beautiful"를 변수 Ugly에 할당함	**Beautiful**	-	-
2	Beautiful = "Ugly"	문자열 "Ugly"를 변수 Beautiful에 할당함	Beautiful	**Ugly**	-
3	Handsome = Ugly	Ugly 변수의 값을 Handsome 변수에 할당함	Beautiful	Ugly	**Beautiful**
4	print("Beautiful")	문자열 "Beautiful"을 출력			
5	print(Ugly)	문자열 "Beautiful"을 출력			
6	print(Handsome)	문자열 "Beautiful"을 출력			

예제 8.1-2 두 변수의 값을 서로 맞바꾸기

다음을 만족하는 파이썬 프로그램을 작성하고 추적표를 만들어라. 사용자로부터 두 개의 숫자값을 입력받아 각각을 변수 a와 b에 할당한다. 마지막에 변수 a와 b를 출력할 때 그 값이 서로 바뀌어 있어야 한다. 예를 들어, 사용자로부터 숫자 5와 7을 입력받아 5는 변수 a, 7은 변수 b에 할당하였다면 프로그램 마지막에 변수 a와 b의 값을 출력하였을 때 변수 a는 7이라는 값, 변수 b는 5라는 값을 출력해야 한다.

풀이

다음 파이썬 프로그램은 두 변수의 값을 맞바꾸는 것처럼 보이지만, 실제로는 그렇지 않다.

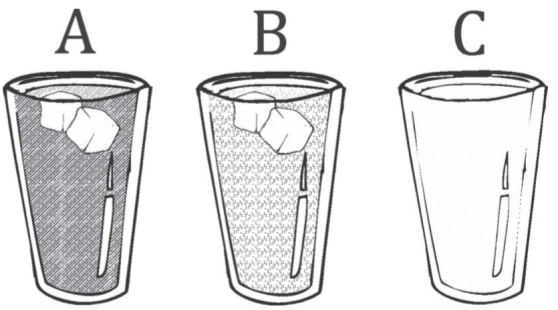
```
a = int(input())
b = int(input())

a = b
b = a

print(a)
print(b)
```

다음 추적표를 참고하여 사용자가 숫자 5와 7을 입력하였을 때 두 변수의 값이 어떻게 바뀌는지 살펴보자.

단계	명령문	설명	a	b
1	a = int(input( ))	사용자가 입력한 숫자 5를 변수 a에 할당	**5**	-
2	b = int(input( ))	사용자가 입력한 숫자 7을 변수 b에 할당	5	**7**
3	a = b	변수 b의 값을 변수 a에 할당	**7**	7
4	b = a	변수 a의 값을 변수 b에 할당	7	**7**
5	print(a)	숫자 7이 출력		
6	print(b)	숫자 7이 출력		

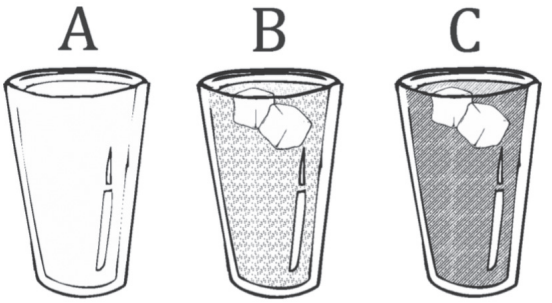
이럴 수가! 숫자 5가 어디로 사라졌을까? 위 파이썬 프로그램은 두 변수의 값을 맞바꾸지 못하였다. 그럼 어떻게 하면 두 변수의 값을 서로 바꿀 수 있을까?

다음과 같이 두 개의 물컵이 있다고 가정해 보자. 물컵 A에는 오렌지 주스가 담겨 있고, 물컵 B에는 레몬주스가 담겨 있다. 두 물컵의 내용물을 서로 맞바꾸려면 비어 있는 물컵 C가 하나 더 있어야 한다.

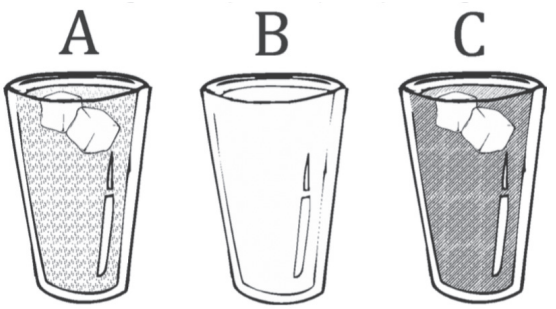


빈 물컵 C가 있을 때 다음 순서로 물컵 A와 B의 내용물을 맞바꿀 수 있다.

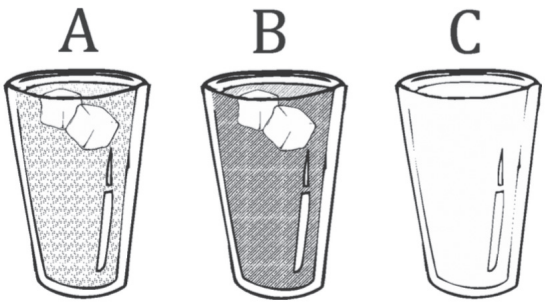
1. 물컵 A에 담겨 있는 오렌지 주스를 물컵 C에 따른다.



2. 물컵 B에 담겨 있는 레몬주스를 물컵 A에 따른다.



3. 물컵 C에 담겨 있는 오렌지 주스를 물컵 B에 따른다.



두 물컵의 내용물이 성공적으로 맞바뀌었다.

이런 방식으로 파이썬 프로그램을 다음과 같이 수정해 보자.

```
a = int(input())
b = int(input())
c = a  # 물컵 A에 담겨 있는 오렌지 주스를 물컵 C에 따른다.
a = b  # 물컵 B에 담겨 있는 레몬주스를 물컵 A에 따른다.
b = c  # 물컵 C에 담겨 있는 오렌지 주스를 물컵 B에 따른다.
print(a)
print(b)
```

주목할 것! 위 파이썬 코드의 할당 명령문 뒤에 "#" 기호가 있는 것에 주목하자. "#" 기호 다음에 있는 문자들은 주석으로 간주되며 실행되지 않는다.

두 변수의 값을 맞바꾸기 위한 파이썬 프로그램을 다음과 같이 작성할 수도 있다.

```
a = int(input())
b = int(input())

a, b = b, a

print(a)
print(b)
```

예제 8.1-3 두 변수의 값을 맞바꾸는 다른 방법

다음을 만족하는 파이썬 프로그램을 작성하고 추적표를 만들어라. 사용자로부터 두 개의 숫자값을 입력받아 각각을 변수 a와 b에 할당한다. 마지막에 변수 a와 b를 출력할 때 그 값이 서로 바뀌어 있어야 한다. 앞서 살펴본 방법 이외에 다른 방법을 사용해야 한다.

풀이

값이 숫자일 때는 다음 파이썬 코드로 두 변수의 값을 맞바꿀 수 있다.

```
a = int(input())
b = int(input())

a = a + b
b = a - b
a = a - b

print(a)
print(b)
```

주목할 것! 위 방식의 단점은 숫자 이외에 알파벳 등의 문자열 값은 맞바꿀 수 없다는 점이다.

예제 8.1-4 추적표 만들기

다음 파이썬 프로그램에 사용자로부터 0.3, 0.45, 10이라는 값이 입력되었을 때의 추적표를 각각 만들어라.

```
b = float(input())
c = 3
c = c * b
a = 10 * c % 10

print(a)
```

풀이

i. 입력값이 0.3인 경우

단계	명령문	설명	a	b	c
1	b = float(input( ))	사용자로부터 0.3 값을 입력받음	-	**0.3**	-
2	c = 3	숫자 3을 변수 c에 할당	-	0.3	3
3	c = c * b	변수 c와 b의 곱을 변수 c에 할당	-	0.3	**0.9**
4	a = 10 * c % 10	변수 c에 10을 곱한 결과를 10으로 나눈 나머지 값을 변수 a에 할당	**9**	0.3	0.9
5	print(a)	9를 출력			

ii. 입력값이 4.5인 경우

단계	명령문	설명	a	b	c
1	b = float(input())	사용자로부터 4.5 값을 입력받음	-	**4.5**	-
2	c = 3	숫자 3을 변수 c에 할당	-	4.5	**3**
3	c = c * b	변수 c와 b의 곱을 변수 c에 할당	-	4.5	**13.5**
4	a = 10 * c % 10	변수 c에 10을 곱한 결과를 10으로 나눈 나머지 값을 변수 a에 할당	**5**	4.5	13.5
5	print(a)	5를 출력			

iii. 입력값이 10인 경우

단계	명령문	설명	a	b	c
1	b = float(input())	사용자로부터 10 값을 입력받음	-	**10**	-
2	c = 3	숫자 3을 변수 c에 할당	-	10	**3**
3	c = c * b	변수 c와 b의 곱을 변수 c에 할당	-	10	**30**
4	a = 10 * c % 10	변수 c에 10을 곱한 결과를 10으로 나눈 나머지 값을 변수 a에 할당	**0**	10	30
5	print(a)	0을 출력			

예제 8.1-5 추적표 만들기

다음 파이썬 프로그램에서 사용자가 3이라는 값을 입력했을 때의 추적표를 만들어라.

```
a = float(input())

b = a + 10
a = b * (a - 3)
c = 3 * b / 6
d = c * c
d -= 1

print(d)
```

풀이

단계	명령문	설명	a	b	c	d
1	a = float(input( ))	사용자로부터 3 값을 입력받음	**3**	–	–	–
2	b = a + 10	변수 b에 변수 a의 값과 10을 더한 결괏값을 할당	3	**13**	–	–
3	a = b * (a – 3)	변수 a에서 3을 뺀 값에 변수 b의 값을 곱한 결과를 변수 a에 할당	**0**	13	–	–
4	c = 3 * b / 6	변수 b의 값에 3을 곱한 값을 6으로 나눈 결괏값을 변수 c에 할당	0	13	**6.5**	–
5	d = c * c	변수 c의 값을 제곱한 결괏값을 변수 d에 할당	0	13	6.5	**42.25**
6	d -= 1	변수 d를 1 감소	0	13	6.5	**41.25**
7	print(d)	41.25를 출력				

8.2 복습문제: 참/거짓

다음 문제를 읽고 **참** 또는 **거짓**으로 답하여라.

1. 추적표는 컴퓨터를 검사하는 기술이다.
2. 추적표는 프로그래머가 컴퓨터 프로그램의 오류를 찾는 데 도움이 된다.
3. 추적표를 먼저 작성하지 않으면 컴퓨터 프로그램을 작성할 수 없다.
4. 두 변수의 값을 서로 맞바꾸기 위해서는 반드시 추가 변수를 사용해야 한다.

8.3 프로그래밍 연습문제

다음 프로그래밍 연습문제를 완성하여라.

1. 다음 파이썬 프로그램에 사용자로부터 (i) 3, (ii) 4, (iii) 1이라는 값이 입력되었을 때의 추적표를 각각 만들어라.

```
a = int(input())

a = (a + 1) * (a + 1) + 6 / 3 * 2 + 20
b = a % 13
c = b % 7
d = a * b * c
print(a, ",", b, ",", c, ",", d)
```

2. 다음 파이썬 프로그램에 사용자로부터 (i) 3, 4, (ii) 4, 4라는 값이 입력되었을 때의 추적표를 각각 만들어라.

```
a = float(input())
b = float(input())

c = a + b
d = 1 + a / b * c + 2
e = c + d
c += d + e
e -= 1
d -= c + d % c
print(c, ",", d, ",", e)
```

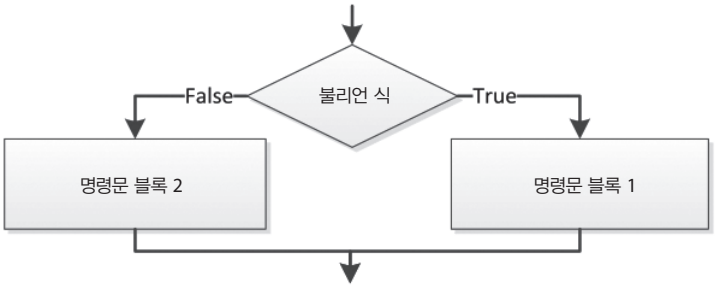
CHAPTER

17

이중-택일 결정 구조

17.1 이중-택일 결정 구조

이중-택일 결정 구조(dual-alternative decision structure)는 명령문 블록이 양쪽 경로(True/False) 모두에 포함되는 형태를 가진다.



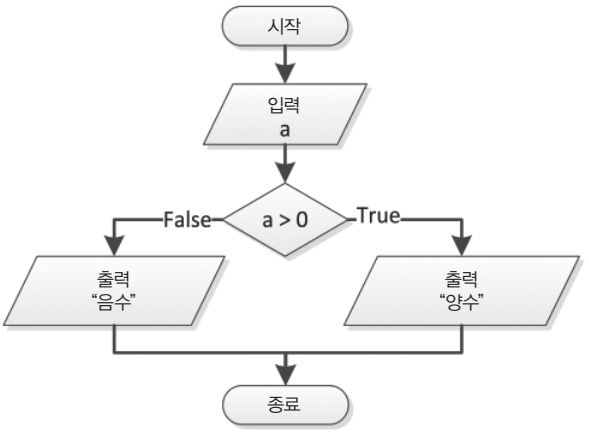
위 그림에서 불리언 식의 결과가 True이면 명령문 블록 1이 수행되고, 그렇지 않은 경우(불리언 식의 결과가 False인 경우)에는 명령문 블록 2가 수행된다.

파이썬에서 일반 형태의 이중-택일 결정 구조는 다음과 같다.

```
if 불리언 식:
    명령문 블록 1
else:
    명령문 블록 2
```

예제 17.1-1 출력 메시지 구하기

다음 순서도에서 입력값이 (i) 3, (ii) -3, (iii) 0인 경우에 각각 출력값이 무엇인지 적어라.



풀이

i. 사용자가 3이라는 값을 입력했다면, 불리언 식의 결과는 True다. 순서도의 수행 흐름이 오른쪽 명령문 블록을 따라가므로 "양수" 값이 출력된다.

ii. 사용자가 -3이라는 값을 입력했다면, 불리언 식의 결과는 False다. 순서도의 수행 흐름이 왼쪽 명령문 블록을 따라가므로 "음수" 값이 출력된다.

iii. 사용자가 0이라는 값을 입력했다면, 불리언 식의 결과는 False다. 순서도의 수행 흐름이 왼쪽 명령문 블록을 따라가므로 "음수" 값이 출력된다.

주목할 것!

분명히 이 순서도는 모든 면에서 완벽하지 않다. 아시다시피 0은 음수도 아니고, 양수도 아니다. 차후에 다루게 될 중첩 결정 제어 구조를 배우면, "입력한 값은 0입니다"라는 세 번째 메시지의 출력 방법을 알게 될 것이다.

기억할 것!

결정 기호는 한 개의 입구와 두 개의 출구를 가진다. 두 개의 출구만을 가지므로 세 가지 경우의 수에 대한 메시지는 출력할 수 없다.

예제 17.1-2 추적표와 이중-택일 결정 구조

다음 파이썬 프로그램에 대하여 입력값이 (i) 5, (ii) 10일 때, 각 단계별 변숫값을 나타내는 추적표를 완성하여라.

file_17_1_2

```
a = float(input())

z = a * 10
w = (z - 4) * (a - 3) / 7 + 36

if a < z >= w:
    y = 2 * a
else:
    y = 4 * a

print(y)
```

풀이

i. 입력값 5에 대한 추적표는 다음과 같다.

단계	명령문	설명	a	z	w	y
1	a = float(intput())	사용자가 5 값을 입력	**5**	-	-	-
2	z = a * 10		5	**50**	-	-
3	w = (z - 4) * (a - 3) / 7 + 36		5	50	**49.142**	-
4	if a < z >= w:	True로 평가됨				
5	y = 2 * a		5	50	49.142	**10**
6	print(y)	10 값이 출력됨				

ii. 입력값 10에 대한 추적표는 다음과 같다.

단계	명령문	설명	a	z	w	y
1	a = float(intput())	사용자가 10 값을 입력	**10**	-	-	-
2	z = a * 10		10	**100**	-	-
3	w = (z - 4) * (a - 3) / 7 + 36		10	100	**132**	-
4	if a < z >= w:	False로 평가됨				

5	`y = 4 * a`		10	100	132	**40**
6	`print(y)`	40 값이 출력됨				

예제 17.1-3 누가 가장 큰가?

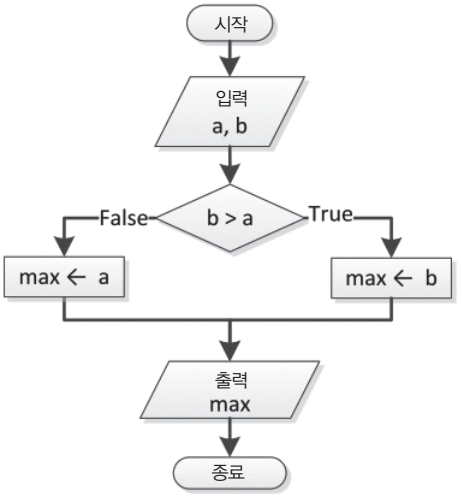
사용자로부터 두 수를 입력받아 변수 a와 b에 저장하고, 둘 중에 큰 값을 출력하는 순서도를 설계하고, 이에 대한 파이썬 프로그램을 작성하여라.

풀이

이 예제는 세 가지 방법으로 해결할 수 있으며, 첫 번째와 두 번째 방법은 각각 이중-택일 결정 구조와 단일-택일 결정 구조를 이용하는 것이다. 세 번째 방법은 파이썬스러운 방식으로 문제를 해결한다.

첫 번째 방법 — 이중-택일 결정 구조 이용하기

이 방법은 b가 a보다 큰지를 검사한다. 그 결과가 True이면 b가 큰 값이고, 그렇지 않으면(결과가 False인 경우) a가 큰 값이다.



file_17_1_3a

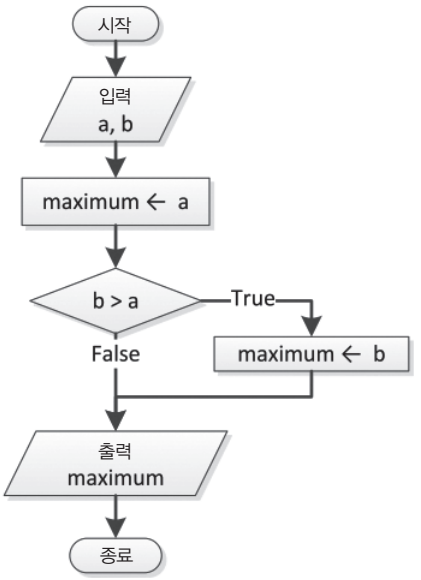
```
a = float(input())
b = float(input())
```

```
if b > a:
    maximum = b
else:
    maximum = a

print("큰 값: ", maximum)
```

주목할 것! 이 예제는 두 수 중에서 누가 더 큰지를 결정하는 것이지 큰 값을 변수 a나 b에 다시 할당하지는 않는다.

두 번째 방법 — 단일-택일 결정 구조 이용하기

이 방법은 초기에 a가 큰 수라고 가정한다. 따라서 변수 a의 값을 maximum 변수에 할당한다. 하지만 변수 a와 b의 값을 비교한 후 b가 더 큰 값으로 판명될 경우, maximum 변수의 값을 변경한다. 즉, 변수 b의 값으로 maximum 변수의 값을 변경한다. 어떠한 상황이 벌어지든 결국 maximum 변수에는 항상 큰 값이 할당된다.



file_17_1_3b

```
a = float(input())
b = float(input())
```

```
maximum = a
if b > a:
    maximum = b

print("큰 값: ", maximum)
```

세 번째 방법 — 파이썬스러운 방식

이 방법은 결정 구조를 사용하지 않고, 가장 단순한 방식으로 파이썬 함수 중 max() 함수를 사용하는 것이다.

file_17_1_3c

```
a = float(input())
b = float(input())

maximum = max(a, b)
print("큰 값: ", maximum)
```

예제 17.1-4 홀수와 짝수 찾기

사용자로부터 정숫값을 입력받은 후, 입력값이 홀수 또는 짝수인지를 출력하는 순서도를 설계하고, 이에 대한 파이썬 프로그램을 작성하여라.

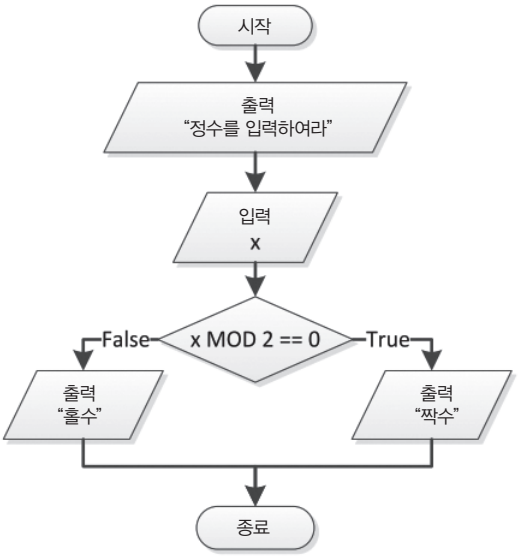
풀이

이 예제를 해결하기 위해서는 먼저 입력된 숫자가 홀수인지 짝수인지를 판단해야 한다. 우선 홀수나 짝수의 일반적 속성을 이해해야 한다. 짝수는 2로 나누어떨어진다. 따라서, 정수 x에 대하여 x % 2 연산을 수행하였을 때의 결과가 0이면, x는 짝수다. 그렇지 않을 경우에는 x는 홀수다.

이제 사용자가 입력한 숫자에 대하여 그 숫자가 홀수인지 짝수인지를 판별할 수 있다. 이런 속성을 이용하여 홀수와 짝수를 판별하였을 때의 결과는 다음과 같다.

- 홀수: 1, 3, 5, 7, 9, 11, …
- 짝수: 0, 2, 4, 6, 8, 10, 12, …

주목할 것! 0은 짝수에 포함된다.



file_17_1_4

```
x = int(input("정수를 입력하여라: "))

if x % 2 == 0:
    print("짝수")
else:
    print("홀수")
```

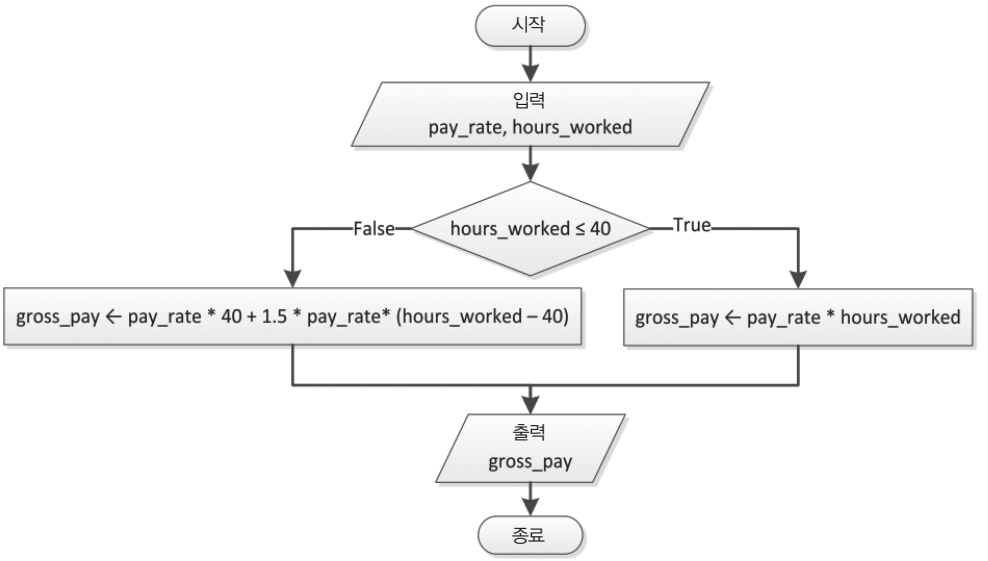
예제 17.1-5 주급 계산하기

일주일의 총 급여(주급)는 시간당 급여와 일주일 동안의 근로 시간에 의해 계산될 수 있다. 그러나 일주일에 40시간 이상을 근로하였다면 40시간 이상의 초과 시간에 대해서는 시간당 급여의 1.5배를 받는다. 사용자로부터 시간당 급여와 일주일 동안의 근로 시간을 입력받고, 일주일의 총 급여를 출력하는 순서도를 설계하고 이에 대한 파이썬 프로그램을 작성하여라.

풀이

이 예제는 이중-택일 결정 구조를 이용하여 해결할 수 있다. 만약, 일주일에 40시간 이상 근로하였다면 일주일의 총 급여는 다음과 같이 구할 수 있다.

*gross_pay = pay_rate * 40 + 1.5 * pay_rate * all_hours_worked_over_40*



file_17_1_5

```
pay_rate = float(input())
hours_worked = int(input())

if hours_worked <= 40:
    gross_pay = pay_rate * hours_worked
else:
    gross_pay = pay_rate * 40 + 1.5 * pay_rate * (hours_worked - 40)

print("급여:", gross_pay)
```

17.2 복습문제: 참/거짓

다음 문제를 읽고 **참** 또는 **거짓**으로 답하여라.

1. 이중-택일 결정 구조의 내부 명령문은 한 번도 수행되지 않을 수도 있다.
2. 이중-택일 결정 구조는 최소 두 개의 명령문을 포함하여야 한다.
3. 이중-택일 결정 구조는 else라는 예약어를 사용한다.

4. 다음 명령문에는 구문 오류가 없다.

```
else = 5
```

5. 이중-택일 결정 구조에서 불리언 식은 두 개 이상의 값을 반환할 수 있다.

6. 다음 코드는 효과성의 속성을 만족한다.

```
x = int(input())
y = int(input())

z = int(input())

if x > y and x > z:
    print(x, "가 큰 수입니다.")
else:
    print(y, "가 큰 수입니다.")
```

17.3 복습문제: 객관식

다음 문제에서 옳은 것을 골라라.

1. 이중-택일 결정 구조는 어떤 경로에 명령문 블록을 포함하는가?

   a. False 경로

   b. True와 False 경로

   c. False 경로

2. 다음 코드에서 y += 1 명령문은 어떤 경우에 수행되는가?

```
if x % 2 == 0:
    x = 0
else:
    y += 1
```

   a. 변수 x가 2로 나누어떨어질 때

   b. 변수 x가 짝수값을 가지고 있을 때

   c. 변수 x가 홀수값을 가지고 있을 때

   d. a, b, c 모두 옳지 않다.

3. 다음 코드에서 y += 1 명령문은 어떤 경우에 수행되는가?

```
if x == 3:
    x = 5
else:
    x = 7
y += 1
```

a. 변수 x의 값이 3일 때

b. 변수 x의 값이 3이 아닐 때

c. a, b 모두 옳다.

17.4 프로그래밍 연습문제

다음 프로그래밍 연습문제를 완성하여라.

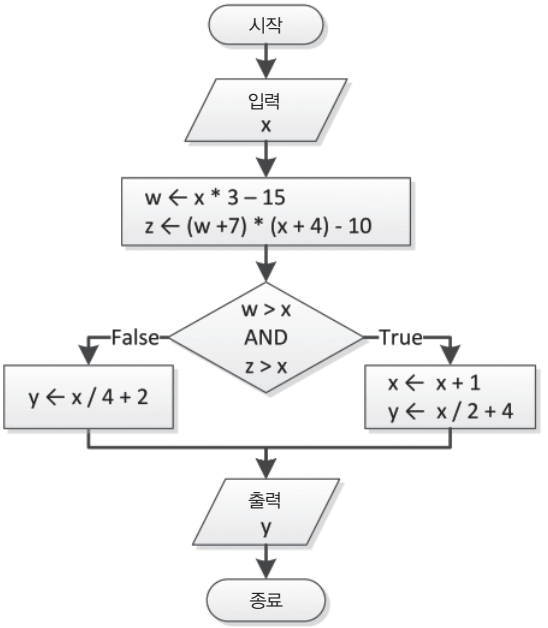
1. 다음 파이썬 프로그램과 일치하는 순서도를 작성하고, 입력값이 (i) 3, (ii) 0.5일 때, 각 단계별 변숫값을 나타내는 추적표를 완성하여라.

```
a = float(input())
z = a * 3 - 2
if z >= 1:
    y = 6 * a
else:
    z += 1
    y = 6 * a + z

print(z, ",", y)
```

2. 다음 파이썬 프로그램과 일치하는 순서도를 작성하고, 각 단계별 변숫값을 나타내는 추적표를 완성하여라.

```
import math

x = 3
y = x ** 3 + 9
z = 2 * x + y - 4
if x > y:
    y = z % x
    z = math.sqrt(x)
else:
    x = z % y
    z = math.sqrt(y)

print(x, ",", y, ",", z)
```

3. 다음 순서도에 대한 파이썬 프로그램을 작성하고, 입력값이 (i) 10, (ii) 2일 때, 각 단계별 변숫값을 나타내는 추적표를 완성하여라.



4. 사용자로부터 정숫값을 입력받고, 입력값이 6의 배수인지 여부를 출력하는 파이썬 프로그램을 작성하여라.
5. 사용자로부터 정숫값을 입력받고, 입력값이 6 또는 7의 배수인 경우와 그렇지 않을 경우(6 또는 7의 배수가 아닌 경우)에 서로 다른 메시지를 출력하는 파이썬 프로그램을 작성하여라.
6. 사용자로부터 정숫값을 입력받고, 입력값이 4의 배수인지 여부를 출력하는 파이썬 프로그램을 작성하여라. 단, 입력값을 4로 나누었을 때 몫과 나머지를 출력해야 한다. 예를 들어, 사용자가 14 값을 입력하였다면 "14 = 4 * 3 + 2"라는 메시지가 출력되어야 한다.
7. 사용자로부터 정숫값을 입력받고, 입력값이 네 자리 정수인지 여부를 출력하는 파이썬 프로그램을 작성하여라.

   힌트: 네 자리 정수는 1000부터 9999까지의 정수다.
8. 사용자로부터 두 개의 정숫값을 입력받고, 입력값 중 작은 값을 출력하는 파이썬 프로그램과 순서도를 작성하여라(단, 서로 다른 정숫값을 사용자가 입력한다고 가정).

9. 사용자로부터 세 개의 정숫값을 입력받고, 이들 입력값을 길이로 하여 삼각형을 만들 수 있는지의 여부를 출력하는 파이썬 프로그램을 작성하여라.

힌트: 삼각형 한 변의 길이는 다른 두 변의 길이 합보다 작다.

10. 사용자로부터 세 개의 정숫값을 입력받고, 이들 입력값을 길이로 하여 직각 삼각형을 만들 수 있는지의 여부를 출력하는 파이썬 프로그램을 작성하여라. 입력값을 3, 4, 5로 하여 작성한 프로그램을 검사해 보라.

힌트: 피타고라스의 정리를 사용한다.

11. 멀리뛰기 시합에 참가하기 위해서는 세 번의 점프를 시도하여 평균 거리가 8미터 이상이어야 한다. 사용자로부터 세 개의 거리 값을 입력받고, 세 값의 평균 거리가 8미터 이상이면 "통과"를 출력하고, 8미터 미만이면 "실패"를 출력하는 파이썬 프로그램을 작성하여라.

12. 일주일의 총 급여는 시간당 급여와 일주일 동안의 근로 시간에 의해 계산될 수 있다. 그러나 일주일에 40시간 이상을 근로하였다면 40시간 이상의 초과 시간에 대해서는 시간당 급여의 2배를 받는다. 사용자로부터 시간당 급여와 일주일 동안의 근로 시간을 입력받고, 일주일의 총 급여 실 수령액을 출력하는 파이썬 프로그램과 순서도를 작성하여라. 실 수령액은 총 급여에서 공제(세금, 건강보험, 퇴직금 등)를 제외한 금액을 말한다. 단, 공제율은 30%라고 가정한다.

13. 자동차의 정기 점검을 통해 사고를 예방하고 자동차를 안정적으로 운행할 수 있다. 예를 들어, 다음과 같이 두 가지 형태의 서비스가 있다고 가정해 보자.

a. 단기 정기 점검 서비스

b. 장기 정기 점검 서비스

단, 단기 정기 점검 서비스는 6,000마일마다 받아야 하며, 장기 정기 점검 서비스는 12,000마일마다 받아야 한다.

사용자로부터 운행한 마일 수를 입력받고, 다음 정기 점검 서비스를 받을 때까지 남은 운행 마일 수와 서비스 종류를 출력하는 파이썬 프로그램을 작성하여라.

14. 두 개의 자동차(자동차 A와 B)가 정지 상태에서 각기 서로 다른 가속도로 일정 시간 동안 직선 도로를 운행하고 있다고 하자. 사용자로부터 두 자동차의 운행 시간(두 자동차의 운행 시간은 동일)과 각 자동차의 가속도값을 입력받고, 두 자동차 간의 거리 차이와 두 자동차 중에 어떤 자동차가 앞서가는지를 출력하는 파이썬 프로그램을 작성하여라.

힌트: $S = u_o + \frac{1}{2}at^2$

CHAPTER

25

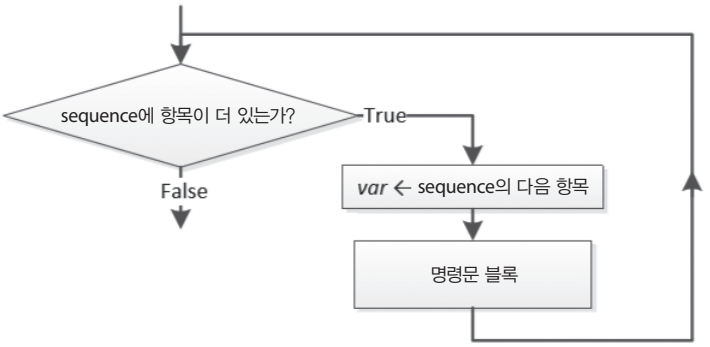
for-루프

25.1 for-루프

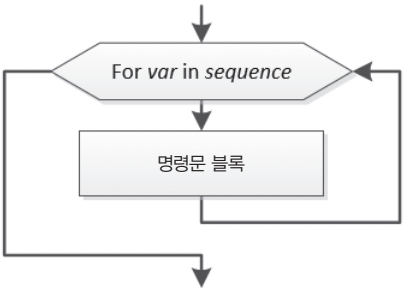
24장에서 사전-검사, 중간-검사, 사후-검사 루프를 이용하여 무한 반복 구조와 유한 반복 구조를 만드는 방법에 대해서 배웠다. 이들 구조는 정해진 횟수만큼 반복을 수행하는 데 유용할 뿐만 아니라 횟수가 정해져 있지 않은 반복을 수행하는 데도 유용하다. 그러나 유한 반복 구조가 컴퓨터 프로그래밍에서 훨씬 더 많이 사용되기 때문에 파이썬을 포함한 거의 대부분의 컴퓨터 프로그래밍 언어에는 while 명령문보다 좀 더 이해하기 쉽고 편리하게 사용할 수 있는 특별한 명령문을 가지고 있다. 그것은 바로 for 명령문이다. for 명령문의 일반 형태는 다음과 같다.

```
for var in sequence:
    명령문 블록
```

여기서 var는 sequence 내부의 값을 연속해서 할당받는 변수이며, 명령문 블록은 var 변수 각각의 값마다 한 번씩 수행된다. 파이썬 for 명령문의 순서도는 다음과 같다.



그러나 위 순서도보다 좀 더 단순한 형태를 가지는 다음 순서도를 앞으로 사용할 것이다.

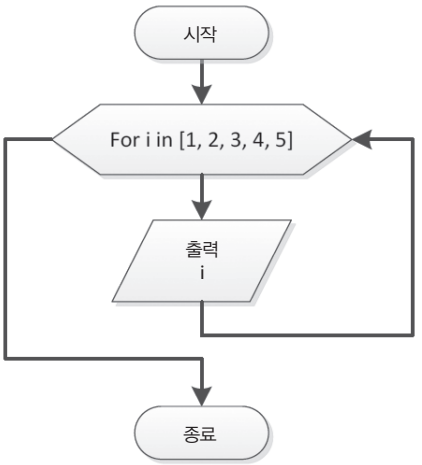


다음 예제를 살펴보자.

file_25_1a

```
for i in [1, 2, 3, 4, 5]:
    print(i)
```

위 예제는 1, 2, 3, 4, 5를 차례대로 화면에 출력한다. 순서도는 다음과 같다.

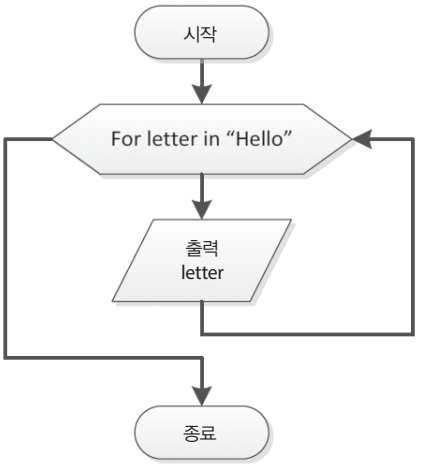


다음 예제를 살펴보자.

file_25_1b

```
for letter in "Hello":
    print(letter)
```

위 예제는 "H", "e", "l", "l", "o" 문자를 차례대로 화면에 출력한다. 순서도는 다음과 같다.



파이썬의 range() 함수는 연속적인 정수를 생성하는 데 사용된다. 이 함수를 for 명령문과 함께 사용하면 다음과 같이 확장할 수 있다.

```
for var in range([initial_value,] final_value [, step ]):
    명령문 블록
```

여기서

- initial_value는 연속적인 정수의 시작값을 나타낸다. 이 인자는 옵션이다. 생략하면 기본값은 0이다.
- final_value는 연속적인 정수의 마지막을 나타낸다. 그러나 final_value는 포함되지 않는다.
- step은 연속한 두 정수 사이의 차이를 나타낸다. 이 인자는 옵션이다. 생략하면 기본값은 1이다.

주목할 것! initial_value, final_value, step 값은 반드시 정수이어야 한다.

다음 예제는 0부터 10까지의 숫자를 차례대로 출력한다.

file_25_1c

```
for i in range(0, 11, 1):
    print(i)
```

물론, step이 1이기 때문에 세 번째 인자는 생략할 수 있다. 그래서 위 예제를 다음과 같이 재작성할 수 있다.

file_25_1d

```
for i in range(0, 11):
    print(i)
```

게다가, 초깃값이 0이기 때문에 첫 번째 인자 또한 생략할 수 있다. 그래서 위 예제를 다음과 같이 재작성할 수 있다.

file_25_1e

```
for i in range(11):
    print(i)
```

마지막으로, sequence의 값을 역순으로 출력하고자 한다면 step 값을 음수로 사용해야 한다. 다음 예제는 10부터 2까지의 짝수를 차례대로 화면에 출력한다.

file_25_1f

```
a = 10
b = 0
for i in range(a, b, -2):
    print(i)
```

var 값을 루프 내부에서 변경하지 않아야 한다. 마찬가지로 initial_value, final_value, step 값 또한 변경하지 않아야 한다. 만일 변경하고자 한다면 for-루프 대신 while-루프를 사용해라.

while-루프의 경우와 마찬가지로, for-루프도 else 키워드와 조합하여 사용할 수 있다.

```
for var in sequence:
     명령문 블록 1
else:
     명령문 블록 2
```

그러나 이런 방식은 실제로 거의 사용하지 않는다. 그래서 더 이상 이 책에서 언급하지 않겠다.

예제 25.1-1 추적표 생성하기

입력값이 1일 때 다음 파이썬 프로그램의 각 단계별 변숫값을 나타내는 추적표를 생성하여라.

```
a = int(input())

for i in range(-3, 5, 2):
     a = a * 3
print(i, ",", a)
```

풀이

추적표는 다음과 같다.

단계	명령문	설명	a	i	
1	a = int(input())		**1**	-	
2	i = -3		1	**-3**	첫 번째 반복
3	a = a * 3		**3**	-3	
4	i = -1		3	**-1**	두 번째 반복
5	a = a * 3		**9**	-1	

6	i = 1		9	**1**	세 번째 반복
7	a = a * 3		**27**	1	
8	i = 3		27	**3**	네 번째 반복
9	a = a * 3		**81**	3	
10	print(i, ",", a)	3, 81이 출력된다.			

예제 25.1-2 추적표 생성하기

입력값이 4일 때 다음 파이썬 프로그램의 각 단계별 변숫값을 나타내는 추적표를 생성하여라.

```
a = int(input())

for i in range(6, a - 1, -1):
    print(i)
```

풀이

다음은 각 단계별 변숫값을 나타내는 추적표를 보여 준다.

단계	명령문	설명	a	i
1	a = int(input())		**4**	**?**
2	i = 6		4	**6**
3	print(i)	6이 출력된다.		
4	i = 5		4	**5**
5	print(i)	5가 출력된다.		
6	i = 4		4	**4**
7	print(i)	4가 출력된다.		

예제 25.1-3 총 반복 횟수 세기

서로 다른 두 개 입력값에 대해 다음 코드를 각각 실행했을 때 각 실행마다 총 반복 횟수를 적어라. 각 실행마다 입력값은 다음과 같다. (i) 6, (ii) 5

```
n = int(input())
for i in range(5, n + 1):
    print(i)
```

풀이

입력값이 6인 경우 range() 함수는 5와 6으로 이루어진 sequence를 생성한다. 그래서 for-루프는 총 2회의 반복을 수행한다. 한편, 입력값이 5인 경우에는 for-루프가 한 번의 반복만을 수행한다.

예제 25.1-4 10개 숫자의 총합 계산하기

사용자로부터 10개의 숫자를 입력받은 후, 이들의 총합을 계산하고 출력하는 파이썬 프로그램을 작성하여라.

풀이

24장에서 배운 while 명령문을 이용한 코드는 다음과 같다.

```
total = 0

i = 1
while i <= 10:
    a = float(input("숫자를 입력하여라: "))
    total = total + a

    i += 1

print(total)
```

다음은 for 명령문을 사용하여 위 코드를 재작성한 코드다.

file_25_1_4

```
total = 0

for i in range(10):
    a = float(input("숫자를 입력하여라: "))
    total = total + a

print(total)
```

예제 25.1-5 0부터 N까지 제곱근 계산하기

사용자로부터 N개 정수를 입력받은 후 0부터 N까지의 제곱근을 각각 계산하고 출력하는 파이썬 프로그램을 작성하여라.

풀이

파이썬 프로그램은 다음과 같다.

file_25_1_5

```
import math

n = int(input("정수를 입력하여라: "))
for i in range(n + 1):
    print(math.sqrt(i))
```

25.2 for-루프의 적용 규칙

for-루프를 이용하여 프로그램을 작성할 때 다음 규칙을 항상 따라야 한다.

- **규칙 1**: for-루프 내부의 명령문에서 var 변수를 사용할 수는 있지만, var 변수의 값을 변경해서는 안 된다. 마찬가지로 initial_value, final_value, step의 경우도 동일하게 적용된다.
- **규칙 2**: step 변수는 0 값을 가지지 않아야 한다. 0으로 설정되면 파이썬은 오류를 출력한다.
- **규칙 3**: initial_value가 final_value보다 작고 step이 음수이면, 루프는 0회의 반복을 수행한다. 다음 예제는 화면에 아무것도 출력하지 않는다.

```
for i in range(5, 9, -1):
    print(i)
```

- **규칙 4**: initial_value가 final_value보다 크고 step이 양수이면, 루프는 0회의 반복을 수행한다. 다음 예제는 화면에 아무것도 출력하지 않는다.

```
for i in range(10, 6):
    print(i)
```

예제 25.2-1 N개 숫자의 평균 계산하기

사용자로부터 N개 정수를 입력받은 후, 이들 정수의 평균을 계산하고 출력하는 파이썬 프로그램을 작성하여라. N 값은 프로그램의 시작 부분에서 사용자로부터 입력받아야 한다. 작성된 프로그램이 명확성을 만족하는지 검사하여라.

풀이

파이썬 프로그램은 다음과 같다.

file_25_2_5

```
n = int(input("입력할 정수의 개수를 입력하여라: "))

total = 0
for i in range(n):
    a = float(input(str(i + 1) + "번째 정수를 입력하여라: "))
    total = total + a

if n > 0:
    average = total / n
    print("평균:", average)
else:
    print("입력된 정수가 없습니다.")
```

사용자가 변수 N 값으로 양수가 아닌 값을 입력하면 for 명령문은 0회의 반복을 수행한다.

사용자가 변수 N 값으로 0을 입력하면 0-나눗셈(division-by-zero) 오류가 발생한다. 이 오류를 방지하기 위해 if n > 0 명령문이 필요하다. 이 명령문에 의해 명확성이 만족된다.

25.3 복습문제: 참/거짓

다음 문제를 읽고 **참** 또는 **거짓**으로 답하여라.

1. for 명령문에서 변수 var는 sequence로부터 값을 연속해서 자동으로 할당받는다.
2. 유한 반복 구조는 반복 횟수가 정해져 있는 경우에 사용한다.
3. 유한 반복 구조에서 루프 내부의 명령문 블록은 최소한 한 번 수행된다.
4. range() 함수에서 initial_value는 final_value보다 클 수 없다.
5. for-루프를 벗어날 때 var 값은 final_value와 동일하다.
6. range() 함수에서 initial_value, final_value, step의 값은 실수가 아니다.
7. step을 0으로 설정하면 루프는 무한 반복을 수행한다.
8. for 명령문에서 변수 var는 루프 내부의 명령문에서 사용될 수 있으나, 그 값은 절대로 변경되지 않아야 한다.
9. for 명령문에서 step은 특정 상황에서 0이 될 수 있다.

10. 다음 코드는 "안녕"이라는 단어를 10회 출력한다.

```
for i in range(1, 10):
    print("안녕")
```

11. 다음 코드는 "안녕"이라는 단어를 항상 출력한다.

```
b = int(input())
for i in range(0, 9, b):
    print("안녕")
```

12. 다음 코드는 명확성을 만족한다.

```
import math

b = int(input())
for i in range(10):
    a = math.sqrt(b) + i
    b *= 2
```

25.4 복습문제: 객관식

다음 문제에서 옳은 것을 모두 골라라.

1. for 명령문을 사용하는 유한 반복 구조에 대한 설명으로 옳은 것은?
   a. (while 명령문을 사용하는) 사후-검사 루프 구조보다 1회의 반복을 더 수행한다.
   b. (while 명령문을 사용하는) 사후-검사 루프 구조보다 1회의 반복을 덜 수행한다.
   c. 위 모두 옳지 않다.
2. for-루프에 관한 설명으로 옳은 것은?
   a. 사용자가 -1 값을 입력할 때까지 반복하여 숫자를 입력하는 문제에 사용된다.
   b. 사용자의 입력값이 final_value보다 클 때까지 반복하여 숫자를 입력하는 문제에 사용된다.
   c. a, b 모두 옳다.
   d. a, b 모두 옳지 않다.
3. for-루프에서 initial_value, final_value, step은 어떤 값이어야 하는가?
   a. 상숫값
   b. 변수
   c. 표현식
   d. 위 모두 옳다.

4. for-루프에서 initial_value, final_value, step가 변수일 때의 설명으로 옳은 것은?

   a. 루프 내부에서 변경될 수 없다.

   b. 루프 내부에서 변경되지 않아야 한다.

   c. 위 모두 옳지 않다.

5. for-루프에서 var 값이 증가될 때 step은 어떤 값을 가져야 하는가?

   a. 0보다 큰 값이어야 한다.

   b. 0과 같은 값이어야 한다.

   c. 0보다 작은 값이어야 한다.

   d. 위 모두 옳지 않다.

6. for-루프에서 var의 초깃값으로 어떤 값을 가져야 하는가?

   a. 반드시 0 값을 가져야 한다.

   b. 0 값을 가질 수 있다.

   c. 음숫값을 가지지 않아야 한다.

   d. 위 모두 옳지 않다.

7. for-루프에서 변수 var는 sequence의 연속값 각각을 언제부터 자동으로 할당하는가?

   a. 각 반복의 시작부터

   b. 각 반복의 끝부터

   c. 자동으로 할당하지 않는다.

   d. 위 모두 옳지 않다.

8. 다음 코드는 "안녕 철수" 메시지를 몇 회 출력하는가?

```
i = 1
for i in range(5, 6):
    print("안녕 철수")
```

   a. 5회

   b. 1회

   c. 0회

   d. 위 모두 옳지 않다.

9. 다음 코드는 "안녕 영희" 메시지를 몇 회 출력하는가?

```
for i in range(5, 5):
    i = 1
    print("안녕 영희")
```

a. 1회

b. 무한 번

c. 0회

d. 위 모두 옳지 않다.

10. 다음 코드는 "안녕 호동" 메시지를 몇 회 출력하는가?

```
for i in range(5, 6):
    i = 1
    print("안녕 호동")
```

a. 무한 번

b. 1회

c. 0회

d. 위 모두 옳지 않다.

11. 다음 코드는 "안녕 호순" 메시지를 몇 회 출력하는가?

```
for i in range(2, 9):
    if i % 2 == 0:
        print("안녕 호순")
```

a. 8회

b. 7회

c. 5회

d. 위 모두 옳지 않다.

12. 다음 코드는 "안녕 길동" 메시지를 몇 회 출력하는가?

```
for i in range(40, 51):
    print("안녕 길동")
```

a. 1회

b. 2회

c. 10회

d. 11회

13. 다음 코드의 출력값은 무엇인가?

```
k = 0
for i in range(1, 7, 2):
    k = k + i
print(i)
```

a. 3

b. 6

c. 9

d. 위 모두 옳지 않다.

14. 다음 코드의 출력값은 무엇인가?

```
k = 0
for i in range(100, -105, -5):
    k = k + i
print(i)
```

a. -95

b. -105

c. -100

d. 위 모두 옳지 않다.

25.5 프로그래밍 연습문제

다음 프로그래밍 연습문제를 완성하여라.

1. 다음 파이썬 프로그램의 각 단계별 변숫값을 나타내는 추적표를 만들어라. 이 파이썬 프로그램은 얼마나 많은 반복을 수행하는가?

```
a = 0
b = 0
for j in range(0, 10, 2):
    if j < 5:
        b += 1
    else:
        a += j - 1
print(a, ",", b)
```

2. 다음 파이썬 프로그램이 서로 다르게 실행되었을 때 각 실행마다 단계별 변숫값을 나타내는 추적표를 만들어라. 각각의 실행마다 입력값은 다음과 같다. (i) 10, (ii) 21

```
a = int(input())
b = a
for j in range(a - 5, a + 1, 2):
    if j % 2 != 0:
        b = a + j + 5
    else:
        b = a - j
print(b)
```

3. 입력값 12에 대해서 다음 파이썬 프로그램의 각 단계별 변숫값을 나타내는 추적표를 만들어라.

```
a = int(input())
for j in range(2, a, 3):
    x = j * 3 + 3
    y = j * 2 + 10
    if y - x > 0 or x > 30:
        y *= 2
    x += 4
    print(x, ",", y)
```

4. 모든 루프가 정확히 5회 반복을 수행하도록 다음 코드의 빈칸을 채워라.

i.
```
for a in range(5, ______ + 1):
    b += 1
```

ii.
```
for a in range(0, ______, 4):
    b += 1
```

iii.
```
for a in range(______, -17, -2):
    b += 1
```

iv.
```
for a in range(-17, -16, ______):
    b += 1
```

5. 사용자로부터 20개의 숫자를 입력받은 후, 이들 숫자의 곱과 평균값을 계산하고 출력하는 순서도와 파이썬 프로그램을 작성하여라.
6. 사용자로부터 각도 개수를 입력받은 후, 0.5도씩 증가시키면서 0도부터 주어진 각도 개수만큼 사인 값을 계산하고 출력하는 파이썬 프로그램을 작성하여라. 2π는 360°다.

7. 네 자릿수를 가진 정수 30개를 입력받은 후, 첫 번째 자릿수값이 5이고 마지막 자릿수값이 3인 정수들의 총합을 계산하고 출력하는 순서도와 파이썬 프로그램을 작성하여라.
8. 사용자로부터 N개 정수를 입력받은 후, 짝수 정수의 총 개수를 출력하는 순서도와 파이썬 프로그램을 작성하여라. N 값은 프로그램의 시작 부분에서 사용자로부터 입력받는다. 모든 정수가 홀수라면, "짝수 정수를 입력하지 않았습니다."라는 메시지가 출력되도록 한다.
9. 사용자로부터 50개 정수를 입력받은 후, 홀수 정수의 평균값과 짝수 정수의 평균값을 계산하고 출력하는 순서도와 파이썬 프로그램을 작성하여라.
10. 사용자로부터 start와 finish라는 두 개의 정수를 입력받은 후, start와 finish 사이의 모든 정수를 출력하는 순서도와 파이썬 프로그램을 작성하여라. 먼저, 프로그램의 시작 부분에서 start 변숫값이 finish 변숫값보다 큰지를 검사해야 한다. 이런 경우가 발생하면 두 값을 서로 맞바꾼다.
11. 사용자로부터 start와 finish라는 두 개 정수를 입력받은 후, start와 finish 사이에 5의 배수인 모든 정수를 출력하는 순서도와 파이썬 프로그램을 작성하여라. 먼저, 프로그램의 시작 부분에서 start 변숫값이 finish 변숫값보다 큰지를 검사해야 한다. 이런 경우가 발생하면 두 값을 서로 맞바꾼다.
12. 사용자로부터 실수와 정수를 입력받은 후, 누승 연산자(**)를 사용하지 않고 첫 번째 숫자가 두 번째 숫자의 승수로 거듭제곱된 결과를 출력하는 파이썬 프로그램을 작성하여라.
13. 사용자로부터 문자열 메시지를 입력받은 후, 이 메시지에 포함된 단어 수를 출력하는 파이썬 프로그램을 작성하여라. 예를 들어, 입력된 문자열 메시지가 "My name is Bill Bouras"이면, "입력된 메시지는 다섯 개의 단어를 포함하고 있습니다."가 출력된다. 한 개의 빈칸 문자마다 단어가 분리된다고 가정한다.

    힌트: 메시지의 총 문자 개수를 반환해 주는 len() 함수를 사용하여라.
14. 사용자로부터 임의의 문자열을 입력받은 후, 단어별 평균 문자 개수를 출력하는 파이썬 프로그램을 작성하여라. 예를 들어, 입력된 문자열이 "My name is Aphrodite Boura"이면, "단어별 평균 문자 개수는 4.4입니다."가 출력된다. 빈칸 문자는 세지 않는다.